Институт Системного Программирования
Российской Академии Наук

На правах рукописи

БУРДОНОВ Игорь Борисович

Теория конформности для функционального тестирования программных систем на основе формальных моделей

специальность 05.13.11 –
математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ

на соискание учёной степени
доктора физико-математических наук

Москва - 2008

Оглавление

Введение

Структура введения:

- Актуальность исследования
- Общая характеристика работы
- Область исследования
- Проблематика
- Цели и задачи исследования
- Методологическая и теоретическая основа исследования
- Научная новизна исследования
- Практическая значимость работы
- Апробация результатов исследования
- Публикации
- Структура и объём работы

Актуальность исследования

История информационных технологий – это, прежде всего, история их проникновения во все сферы деятельности человека: экономику, военную область, науку, образование, культуру и т.д. Они становятся всё более неотъемлемой частью инфраструктуры современной жизни. Этот процесс, естественно, вызывает интенсивное развитие самих этих технологий: создаются всё более мощные компьютеры и компьютерные системы, всё более сложные программы и программные комплексы.

Однако, чем шире распространяется использование компьютеров, чем более важные и ответственные задачи возлагаются на их «плечи», тем выше становится цена ошибок в программно-аппаратных средствах. Только в США потери от этих ошибок составляют от 20 до 60 млрд. долларов ежегодно, причём около 60% убытков терпят конечные пользователи. Складывается ситуация, когда производители выпускают, а потребители вынуждены

оплачивать заведомо бракованный товар. Но дело не только в материальном ущербе: некорректное поведение компьютерных систем способно вызвать дезорганизацию управления государством, привести к техногенным и экологическим катастрофам, поставить под угрозу жизнь и здоровье людей.

Как показывает практика, чаще всего источником ошибок в компьютерных системах оказывается программное обеспечение (ПО), которое модифицируется гораздо чаще, чем аппаратура. Поэтому проблема повышения качества ПО становится всё более важной и актуальной. Программистские компании вынуждены тратить дорогостоящее время и не менее дорогие усилия разработчиков не только на то, чтобы создать нечто новое, но и гарантировать, что оно действительно решает нужные задачи, делает это правильно, надёжно и безопасно.

Ключевой частью решения этой проблемы является тестирование программ [124], часто требующее времени и ресурсов больше, чем создание этих программ [39-44,62,68]. Из второстепенной и как бы «побочной» деятельности тестирование выходит на первый план, интегрируется с самим процессом разработки ПО и существенно изменяет представления о том, как надо создавать сложные программные комплексы. Тестирование становится жёсткой «обратной связью», замыкает «жизненный цикл» разработки ПО и перестраивает пространственно-временную структуру процесса разработки и организацию коллектива разработчиков.

Противоречивая задача повышения качества программ и снижения затрат на тестирование диктует необходимость поиска путей автоматизации тестирования. Одним из важнейших методов решения этой задачи является тестирование на основе формальных (математических) моделей, целью которого является проверка того, что реализационная модель (модель тестируемой системы) соответствует (конформна) спецификации (модель требований). Конформность понимается как отношение «похожести» реализации на спецификацию. Если спецификационная модель и конформность

определены строго формально, то становится возможным по спецификации автоматически генерировать тесты, проверяющие эту конформность.

В то же время внедрение в практику соответствующих методов тестирования сталкивается сегодня с двумя серьёзными препятствиями. Первое – это большое разнообразие типов моделей и, что ещё важнее, видов конформностей. Для многих конформностей в последние годы были созданы соответствующие математические теории, на основе которых разработаны методы генерации тестов и построены практически используемые тестовые системы. И этот процесс далёк от завершения: продолжает расти количество предлагаемых конформностей и основанных на них методов и инструментов тестирования. В результате тестировщик оказывается перед серьёзной проблемой выбора того, что ему подходит.

Причина такого экстенсивного роста не только, и не столько, во внутренней логике развития теории тестирования, сколько в потребностях практики: «спрос превышает предложение». Поэтому каждый раз, когда практическое тестирование сталкивается с проблемой, которая «не вписывается» в уже существующие теоретические подходы, поиск путей её решения приводит к новому пониманию «похожести» реализации на спецификацию, «изобретению» новой конформности и созданию основанных на ней теории, методов и инструментов тестирования. При этом каждый автор, предлагающий свою конформность, формально начинает теорию «с нуля», и больше внимания уделяет тому, чтобы показать адекватность своей теории той практической проблеме, которая послужила стимулом к её созданию, чем поиску идей и методов, общих с другими теориями конформности.

Вместе с тем сегодня уже становится ясно, что такие общие идеи и методы «витают в воздухе», поскольку в математических формализмах различных конформностей и методах тестирования на их основе много общего. Поэтому задача выявления этих общих идей и методов, задача создания обобщающей теории конформности, покрывающей широкий класс практически

используемых или требуемых конформностей, стала чрезвычайно актуальной как для теории, так и для практики, и впервые наметились пути её решения.

Среди всех видов тестирования наиболее важным для практики является функциональное тестирование. Это тестирование методом «чёрного ящика», когда внутреннее устройство реализации, её внутренние работа и состояния не наблюдаемы, и о правильности или ошибочности реализации судят по её внешнему поведению в ответ на внешние тестовые воздействия. Именно набор этих тестовых воздействий и возможных наблюдений над ответным поведением реализации может быть различным в различных практических случаях. Этот набор определяет ту или иную семантику тестового взаимодействия, которая, в свою очередь, определяет ту или иную конформность. Для программных систем чаще всего используется отношение редукции – «сводимости» реализации к спецификации, когда спецификация предоставляет на выбор несколько вариантов правильного поведения, а правильная программа демонстрирует лишь некоторые из них.

Второе серьёзное препятствие, с которым сталкивается тестирование на основе формальных моделей, – это неразвитость теории тестирования сложных, иерархически построенных систем, состоящих из многих компонентов. Помимо ошибок в реализациях компонентов, здесь появляется новый вид архитектурных ошибок, вызванных тем, что требования к системе некорректно декомпозированы в требования к её компонентам. Такие ошибки не только труднее обнаруживать, но они имеют более печальные последствия: изменение архитектуры системы и переделка всех или части её компонентов. В существующей теории тестирования проблема верификации декомпозиции системных требований не имеет общего решения. Предлагаются лишь частичные решения этой проблемы для некоторых конформностей.

Таким образом, актуальной задачей является создание теории конформности («сводимости» реализации к спецификации при функциональном тестировании), которая предлагала бы как методы автоматической генерации тестов по спецификации, так и методы

автоматической верификации декомпозиции системных требований для сложных систем, состоящих из многих компонентов.

Решению этой задачи посвящена данная диссертация.

Общая характеристика работы

В работе рассматривается класс конформностей типа редукции, основанных на наблюдаемом поведении и параметризуемых семантикой тестового взаимодействия при функциональном тестировании. Такая семантика определяется набором допустимых тестовых воздействий и возможных ответных наблюдений над поведением системы. Такие наблюдения бывают двух типов: наблюдение внешнего действия, выполняемого реализацией, и, в некоторых случаях, наблюдение отказа – отсутствия действий. В качестве основной модели в диссертации выбрана система помеченных переходов (Labelled Transition System – LTS).

Для того, чтобы каждое тестовое воздействие завершалось наблюдением, не должно возникать ненаблюдаемых отказов и дивергенции («зацикливание») – бесконечной внутренней работы системы без взаимодействия с тестом. Кроме этого, в диссертации впервые вводится понятие разрушения, моделирующего поведение, которого не должно быть при тестировании по тем или иным причинам. Предлагается новая концепция безопасного тестирования, которое избегает ненаблюдаемых отказов, дивергенции и разрушения. Вводится гипотеза о безопасности, определяющая класс безопасно-тестируемых реализаций, и основанная на ней безопасная конформность. Для этой конформности предлагается метод генерации безопасных тестов по спецификации, параметризуемый семантикой тестового взаимодействия. Этот метод является обобщением методов генерации тестов для частных случаев, известных в теории и применяемых на практике. Для этих частных случаев он даёт те же результаты, но область его применения гораздо шире.

Также в диссертации рассматривается асинхронное тестирование (тестирование в контексте), когда взаимодействие теста и реализации

происходит не напрямую, а через ту или иную промежуточную среду взаимодействия. Проблема несохранения конформности заключается в том, что асинхронные тесты могут ловить ложные ошибки, то есть те, которые не обнаруживаются при синхронном тестировании, когда тест и реализация взаимодействуют непосредственно друг с другом без какой-либо буферизации в среде. Это является особым случаем общей проблемы композиции, когда составная система, собранная из реализаций компонентов, конформных своим спецификациям, оказывается неконформной композиции этих спецификаций. Тесты, сгенерированные по такой композиции спецификаций, также могут ловить ложные ошибки.

В диссертации предлагается общее решение проблемы несохранения конформности при композиции, в том числе, при асинхронном тестировании с различными средами взаимодействия. Решение основано на специальном алгоритмическом преобразовании спецификаций. Преобразованная спецификация эквивалентна исходной спецификации в том смысле, что определяет тот же класс конформных реализаций и не суженный класс безопасно-тестируемых реализаций. Для преобразованных спецификаций конформность уже сохраняется при композиции и при асинхронном тестировании.

Решение проблемы композиции впервые даёт полное решение проблемы верификации декомпозиции системных требований в рассматриваемой области. Спецификация составной системы оказывается согласованной со спецификациями компонентов тогда и только тогда, когда ей конформна композиция преобразованных спецификаций. Это можно проверить после преобразования спецификаций компонентов и их алгоритмической композиции с помощью методов, основанных на генерации тестов по заданной спецификации системы.

В практическом плане результаты диссертационной работы сводятся к предлагаемым методам генерации тестов и преобразования спецификаций. Это даёт, во-первых, возможность создавать тестовые системы для самых разных

семантик тестового взаимодействия на базе единого подхода с использованием общего метода генерации тестов. Во-вторых, в сочетании с преобразованием спецификации метод применим для асинхронного тестирования с самыми разными средами взаимодействия. В-третьих, заложена основа для построения инструментов автоматической верификации декомпозиции системных требований, причём спецификации компонентов и системы в целом могут пониматься в самых разных семантиках тестового взаимодействия.

Область исследования

Тестирование на основе формальных моделей

Тестирование на основе формальных моделей – это новая дисциплина, сложившаяся последние десятилетия на стыке теории и практики информационных технологий. Это актуальное направление во многом 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 к этой области, появились еще в конце 60-х – начале 70-х годов [30,78,97,123]. Тем не менее, серьезный практический и массовый интерес к результатам в этой области возник только на рубеже веков, в связи с востребованностью эффективных методов контроля и обеспечения качества сложных программных и программно-аппаратных систем в индустрии. Тестирование на основе моделей начинает применяться в промышленных компаниях. Первые же успехи (и, в не меньшей степени, первые неудачи) сделали разработку формальных методов тестирования чрезвычайно актуальными на сегодняшний день [49,50,65,72,75-77,79,80,83,85,87,92,94,106,108,110,114,116,129,135,145,147,151].

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 удаётся строить по частям (инкрементально), постепенно отражая в модели набор свойств реальной системы и, соответственно, расширяя охват тестами различных аспектов работы системы. Как возможность такой инкрементальной разработки моделей и тестов, так и конкретные техники, используемые при этом, существенно зависят от самого *вида* применяемых моделей, и от того, какие имеются *возможности по управлению* тестируемой системой и *наблюдению* ее поведения.

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

Тестирование на основе моделей проверяет соответствие тестируемой системы требованиям. Предполагается, что требования к системе выражены в терминах её взаимодействия с внешним миром (окружением), что как раз и даёт возможность проверять их выполнение в тестовом эксперименте. На формальном уровне соответствие системы требованиям означает, что модель тестируемой системы и модель требований связаны заданным математическим соответствием (бинарным отношением) (Рис.1) [141].

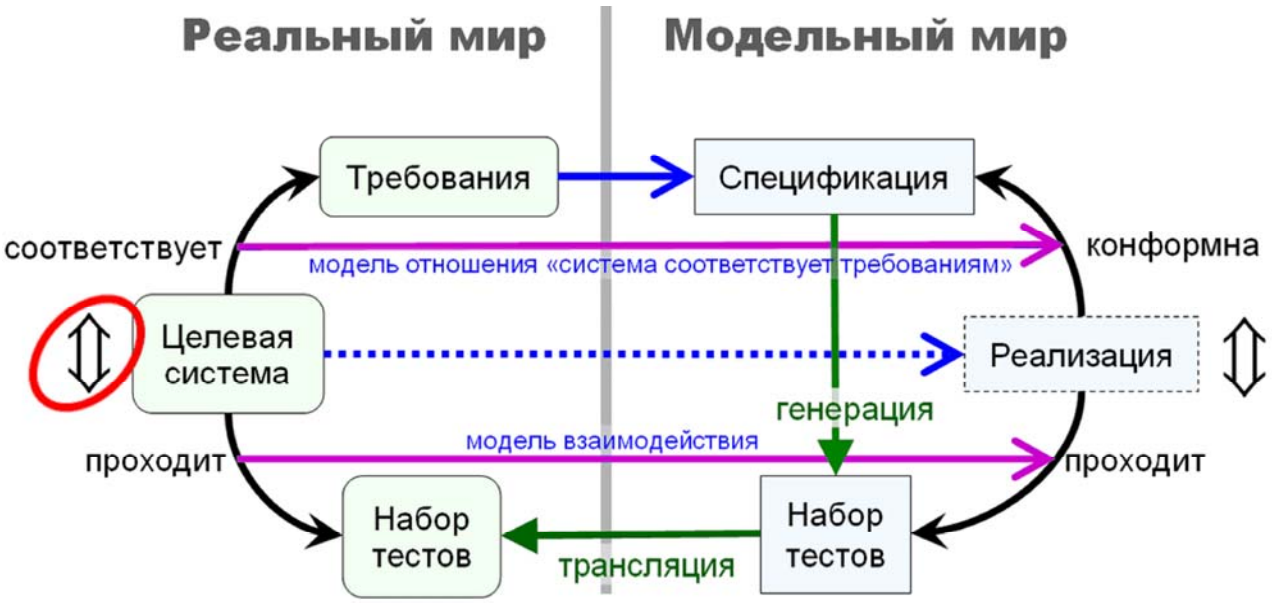


Рис.1.

В диссертационной работе модель тестируемой системы называется реализационной моделью или (для краткости) *реализацией*, модель требований – *спецификацией*, а их соответствие – (модельной) *конформностью*[1] [132,141-143,148]. При тестировании (в отличие от аналитической верификации) предполагается, что реализационная модель не задана или слишком сложна для анализа. Поэтому основной задачей является генерация по заданной спецификации модельных тестов, назначение которых – проверить конформность любой реализационной модели из рассматриваемого класса. Эта проверка основана на модели взаимодействия реальной системы с её окружением. Такую модель, как сказано выше, называют *семантикой взаимодействия*. Вводится модельное отношение «реализация проходит тест». Набор тестов принято называть полным, если реализация проходит каждый тест из набора тогда и только тогда, когда она конформна спецификации. После

[1] Это соответствует общепринятым терминам в англоязычной литературе: *implementation*, *specification* и *conformance*. Отметим, что в русскоязычной литературе до сих пор не сложилось единой терминологии. Слово «соответствие» применяется несколько чаще, чем слово «конформность», но в русском языке оно имеет более широкое смысловое поле, что иногда приводит к недоразумениям. Более того, английское словосочетание "*conformance testing*" в текстах на русском языке встречается чаще, чем все варианты русских словосочетаний (по данным информационно-поисковой системы *Google*).

того, как тесты созданы, они транслируются в реальные тестовые программы, которые, взаимодействуя с реальной системой, устанавливают её соответствие или несоответствие исходным требованиям. Эта схема работы основана на следующих предположениях: во-первых, адекватны модели объектов (реализация и спецификация) и отношений (конформность и семантика взаимодействия), во-вторых, правильно выполняются преобразования (генерация модельных тестов и их трансляция в реальные тестовые программы).

Спецификация неоднозначно определяет реализацию, оставляя разработчику значительную свободу в рамках специфицированных требований. Как именно нужно понимать эти требования, в чём разработчик свободен, а в чём «скован» требованиями, как раз и описывается конформностью. Оно определяется предполагаемыми *правилами взаимодействия* системы с окружением, которые должны быть адекватно отражены на уровне моделей. Спецификация не требует от реализации правильного поведения при взаимодействии с любым её окружением, а только с таким, которое само ведёт себя правильно. Например, в терминах пред- и постусловий [119] тесты должны проверять выполнение постусловия (программа работает правильно) только при таком взаимодействии, которое не нарушает предусловий (к программе правильно обратились). При тестировании правила взаимодействия как раз и определяют те тестовые возможности по управлению и наблюдению за тестируемой системой, которые необходимы и достаточны для проверки конформности.

Выбор модели

По поводу вида моделей, которые нужно использовать для описания сложных систем, большинство исследователей сегодня еще не пришли к общему мнению [24]. Можно выделить три вида моделей в порядке их усложнения.

Самая простая модель – математическая функция как абстракция реентерабельной процедуры. Основная проблема здесь – большое количество

значений аргумента, которые нужно проверить для того, чтобы убедиться, что результат всегда правильный. Особенно, когда аргумент – это большие данные со сложной структурой как, например, для компиляторов.

Вторая модель – это автомат (*finite automata*, *finite state machines*), взаимодействие с которым сводится к той же простой схеме «стимул-реакция», что и для функции. В отличие от функции результат зависит не только от аргумента, но и от состояния автомата, которое является абстракцией таких объектов как глобальные переменные или поля данных объектов классов. Здесь добавляется проблема перебора состояний. Сложность в том, что состояние может быть не наблюдаемо при тестировании.

Третья модель – это система помеченных переходов – LTS (*Labelled Transition System*). Для таких систем характерно сложное взаимодействие. Например, в ответ на стимул может быть несколько реакций или ни одной. Можно подавать несколько стимулов подряд или получать реакции без стимулов. Кроме того, система может выполнять внутренние действия, не наблюдаемые извне. В этих случаях, когда взаимодействие с системой сводится к обмену сообщениями: стимулами и реакциями, такая система называется *реактивной* [76,117,136,149]. В общем случае говорят просто о внешних действиях, наблюдаемых при взаимодействии.

Для LTS характерны такие явления как недетерминизм, отказы, когда нет никаких действий, а также дивергенция – зацикливание, когда система выполняет только внутреннюю работу и не взаимодействует с тестом. В данной работе внимание концентрируется, прежде всего, на этом.

Для систем последнего типа применяются и другие модели: сети Петри, алгебраические спецификации, в частности ASM (*Abstract State Machine*), а также контрактные спецификации, построенные на пред- и постусловиях.

Можно назвать три причины, по которым в данной работе выбрана модель LTS:

- Распространённость этой модели.
- Удобство генерации тестов по LTS.
- На LTS определена композиция, моделирующая взаимодействие компонентов составной системы.

Взаимодействие между LTS-моделямиS понимается как синхронное: передатчик передаёт сообщение, а приёмник принимает его без какой-либо буферизации. Это моделируется оператором параллельной композиции той или иной алгебры процессов. В диссертации выбрана композиция $|\downarrow|$ в духе алгебры процессов CCS (Calculus of Communicating Systems), предложенной Милнером [120,122]. Выдача сообщения `x` обозначается как `!x`, а приём – как `?x`; внутренние действия обозначаются символом τ.

Функциональное тестирование и конформность типа редукции

Что касается конформности, которую следует выбирать для построения тестов, то на этот счёт имеются десятки различных точек зрения. В 1990-93 г.г. Ван Глаббек [89,90] сумел классифицировать для LTS многие из конформностей, описав 155 семантик тестового взаимодействия, на которых они основаны. При этом он не учитывал семантики, специфические для реактивных систем, когда принципиально различаются передача стимулов и передача реакций.

В диссертационной работе исследуется тестирование, которое основано только на наблюдаемом поведении системы. Такое тестирование называют тестированием «чёрного ящика»: тест может только извне взаимодействовать с системой, осуществляя тестовые воздействия и наблюдая внешнее поведение системы, но «не знает и не видит» её внутреннего устройства, отражаемого в понятии состояния, и её внутреннего поведения, не наблюдаемого извне. Такое тестирование называют также *функциональным*, акцентируя внимание на том, что оно проверяет только те свойства системы, которые отражаются в её внешней функциональности и наблюдаемы при её внешнем

функционировании. Понятно, что такое тестирование имеет важное практическое значение, поскольку во многих реальных случаях внутренние действия и состояния реализации не наблюдаемы из теста (например, при удалённом взаимодействии). Но более важно то, что учёт состояний и внутреннего поведения может оказаться «переспецификацией», требуя от реализации того, что никак не проявляется при взаимодействии с ней.

При тестировании методом «чёрного ящика» мы можем наблюдать только такое поведение реализации, которое, во-первых, «спровоцировано» тестом (управление) и, во-вторых, наблюдаемо во внешнем взаимодействии. Такое взаимодействие моделируется с помощью, так называемой, машины тестирования (Милнер [121] и Ван Глаббек [89,90]). Она представляет собой «чёрный ящик», внутри которого находится реализация. Управление сводится к тому, что оператор машины, выполняя тест, нажимает кнопки, «приказывая» или «разрешая» реализации выполнять те или иные действия, которые могут им наблюдаться. Наблюдения (на «дисплее» машины) бывают двух типов: наблюдение некоторого *внешнего действия*, разрешённого оператором и выполняемого реализацией, и наблюдение *отказа* как отсутствия каких бы то ни было действий из числа тех, что разрешены нажатыми кнопками. Важно отметить, что отказ принципиально не сводится к какому-либо внешнему действию, вроде реакции «нет действий», поскольку, в отличие от действия, гарантированно не меняет состояния реализации.

Результатом тестового эксперимента является трасса (последовательность) наблюдений: действий и отказов. В соответствующих конформностях, тем самым, не учитываются состояния. Поэтому в диссертации не рассматриваются конформности типа *симуляций*, которые основаны на соответствии состояний реализации и спецификации.

Также не рассматриваются конформности типа *эквивалентностей*, когда внешнее поведение реализации и спецификации совпадают. В диссертационной работе внимание сосредоточено на конформностях типа *редукции* (сводимости) реализации к спецификации [71]. Суть такой конформности в том, что

конформная реализация не должна демонстрировать внешнее поведение, которое так или иначе запрещено спецификаций, но спецификация может предоставлять на выбор несколько вариантов правильного поведения, и конформная реализация может демонстрировать лишь некоторые из них. Разумеется, речь идёт о поведении в реализации и спецификации в одной и той же ситуации, то есть после одной и той же трассы наблюдений.

К рассматриваемому классу семантик и конформностей относятся: трассовая семантика (*trace semantics*) и трассовый предпорядок (***tr*** – *trace preorder*); семантика завершённых трасс (*completed trace semantics*) и предпорядок завершённых трасс (***ct*** – *completed trace preorder*); семантика трасс с отказами (*failure trace semantics* или *refusal semantics*) и предпорядок трасс с отказами (*failure trace preorder* или ***rf*** – *refusal preorder*); семантика для реактивных систем без блокировки стимулов, но с возможностью наблюдения отсутствия реакций, и конформности ***ior*** (*input-output refusal relation* или *repetitive quiescence relation*) и ***ioco*** (*input-output conformance*); семантика для реактивных систем с блокировками стимулов и конформность $\boldsymbol{ioco}_{\beta\delta}$ (модификация конформности ***ioco***, учитывающая блокировку стимулов); семантика для систем с мультиканалами стимулов и реакций (*MIOTS – Multi Input-Output Transition Systems*) и конформность ***mioco*** (модификация конформности ***ioco*** для таких систем) и др.

Как дальнейшее развитие теории тестирования, так и её практическое применение сталкиваются с рядом проблем, решению которых посвящена диссертационная работа.

Проблематика

Выбор семантики взаимодействия и генерация тестов

Прежде всего, на практике приходится сталкиваться с разнообразием семантик тестового взаимодействия, которое строится на пересечении того, что мы можем, и того, что нам нужно. Речь идёт о тестовых возможностей: какие

есть тестовые воздействия и какие есть наблюдения. То, что мы можем или не можем, – это ограничение тестовых возможностей «сверху». То, что нам требуется для проверки конформности, – это ограничение «снизу».

Отношение ***ioco*** (*Input-Output Conformance*), предложенное Яном Тритмансом [148,149], является одной из наиболее распространённых, хорошо проработанных в теории и зарекомендовавших себя на практике конформностей типа редукции для реактивных систем. Тестовые воздействия двух типов: можно либо послать в реализацию один выбранный стимул, либо принять из реализации одну, но любую из выдаваемых ею реакций. Если тест принимает реакции, он всегда что-нибудь наблюдает: либо реакцию, либо отсутствие реакций (например, по тайм-ауту). Отсутствие реакций называется *стационарностью* (*quiescence*). Если тест посылает стимул, он может наблюдать только приём стимула реализацией. Соответствующий отказ, когда реализация отвергает стимул, называется *блокировкой стимула* (*input refusal*) и считается ненаблюдаемым. Это ограничение «сверху» на наши тестовые возможности.

Для ***ioco*** создано несколько инструментов для генерации тестов, в том числе: TGV (Test Generation with Verification technology) интегрированный в CADP (Construction and Analysis of Distributed Processes Software Tools for Designing Reliable Protocols and Systems), TestGen (входной язык LOTOS, генерирует тест для верификации hardware design и компонентов), TorX (генерация тестов, выполнение их и анализ результатов тестирования on-fly).

В то же время существует целый ряд практических системы, для тестирования которых ***ioco***-семантики недостаточно. Приведём ряд примеров.

Блокировка стимулов обычно (в частности, для ***ioco***) считается ненаблюдаемой (ограничение «сверху»). Но в последние годы появляется всё больше и больше работ, в которых наблюдаемые блокировки допускаются или допускаются частично [21-23,25-27,95,96,115]. Более важно то, что наблюдение блокировок требуется (ограничение «снизу»), например, для тестирования сред взамодействия ограниченной ёмкости, в частности, очередей ограниченной

длины. Здесь стимул – это помещение в очередь, а реакция – выборка из очереди. Когда среда полностью заполнена, стимул должен блокироваться и это нужно уметь проверять.

Другим примером наблюдения блокировок могут служить системы с графическим интерфейсом. Широко распространена ситуация, когда в окне высвечивается меню, в котором некоторые кнопки «бледные», то есть соответствующие стимулы блокируются. Это существенно отличается от приёма стимула с последующей выдачей соответствующей реакции, когда кнопка не «бледная», а в ответ на её нажатие всплывает окошко с надписью «операция не может быть выполнена». Ещё один пример – автомат по продаже «чего-нибудь», который блокирует приём монет, когда это «чего-нибудь» кончается. Это также отличается от автомата, в котором монета в любом случае принимается, но потом возвращается в окошке для сдачи.

Обычно (в частности, для ***ioco***) тест посылает в реализацию стимулы по одному. Необходимость (ограничение «снизу») в передаче сразу нескольких стимулов, из которых реализация сама выбирает один стимул для приёма, требуется для тестирования систем с несколькими входными портами, когда спецификация разрешает любой порядок приёма из портов (система **F** на Рис.2).

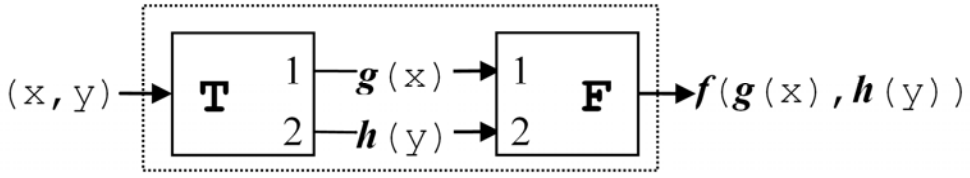


Рис.2.

После передачи одного стимула, тест предлагает реализации оставшиеся стимулы и так далее до тех пор, пока не будут переданы стимулы на все входные порты реализации. Этот пример представляет класс пороговых систем, которые сначала принимают в любом порядке несколько стимулов на разные входные порты, а потом выдают реакцию.

Такая система-приёмник может получать стимулы от другой системы-передатчика, в которой, наоборот, имеется несколько выходных портов и для которой передаваемые стимулы являются реакциями (система **T** на Рис.2). Чтобы обеспечить передачу всех сообщений при любом порядке их приёма, мы вынуждены (ограничение «снизу») потребовать от передатчика того же, что требуем от теста приёмника: начинать с выдачи сразу всех сообщений. Выдача двух реакций последовательно: сначала по первому порту, а потом по второму, или наоборот, является ошибкой. Если тест, как это чаще всего бывает (в частности, для ***ioco***), принимает любую реакцию, то по какому бы порту ни пришла реакция, это приходится считать правильным, и ошибка не обнаруживается. Чтобы обнаружить ошибку, нам нужно уметь в тесте принимать не все реакции, а только из одного порта: один тест обнаружит ошибку как отсутствие реакций по первому порту, а другой – по второму. Та же самая тестовая возможность требуется для тестирования широковещания: для каждого выходного порта нужно проверить, что тест, принимающий реакции из этого порта, не обнаружит их отсутствие. Отношение ***mioco*** [95,96] является как раз такой модификацией отношения ***ioco***, в которой реакции принимаются по отдельным выходным портам системы с возможностью наблюдения частичной стационарности

Совмещение передачи стимулов с приёмом реакций, обычно являющееся излишним (не увеличивающим мощность тестирования), необходимо (ограничение «снизу») для тестирования систем с откатами (Рис.3).

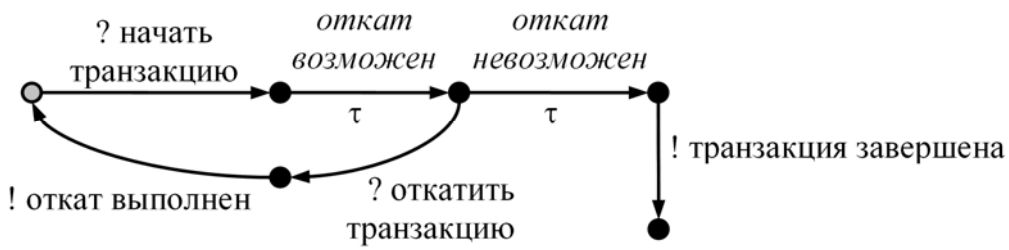


Рис.3.

В такой системе два типа стимулов: «начать транзакцию» и «откатить транзакцию», и, соответственно, два типа реакций: «транзакция завершена» и «откат выполнен». После начала транзакции реализация часто выполняет её в два этапа так, что откат возможен только на первом этапе, а на втором этапе стимул «откатить» отвергается, то есть блокируется. Если блокировки ненаблюдаемы (ограничение «сверху»), мы не можем давать команду «откатить» и, тем самым, не можем тестировать интересующие нас реализации. Однако мы можем обойти это ограничение «сверху», если одновременно с посылкой стимула «откатить» будем принимать реакции. Отказа не будет, так как всегда либо будет принят откат, либо выдана реакция «транзакция завершена».

Поведение, не наблюдаемое при тестировании

Кроме наблюдаемого поведения (действия и отказы) реализация может иметь поведение, которого следует избегать при тестировании по тем или иным причинам. Можно выделить три типа такого поведения.

Первый тип – это ненаблюдаемый отказ. Его возникновение плохо тем, что мы не получим никакого наблюдения в ответ на тестовое воздействие: действий нет, а отказ не наблюдаем. Обычно предполагается, что в тестируемой реализации вообще нет ненаблюдаемых отказов. Для отношения ***ioco*** реализация должна принимать стимулы в каждом своём состоянии (*input-enabled*). В то же время такое требование слишком сильное.

Например, на Рис.4 приведена системы A, взаимодействие с которой происходит по «автоматному» правилу «стимул-реакция». Спецификация – LTS A_0. Формально, она не может служить своей собственной тестируемой реализацией, так как в ней есть блокировка стимула. В то же время тесты, построенные для ***ioco***, не только не вызовут блокировку стимула (после наблюдения трассы стимул не посылается, если его нет в спецификации после этой трассы), но и не обнаружат никаких ошибок. Эти тесты вообще не отличат LTS A_0 от тестируемой и конформной LTS A_1.

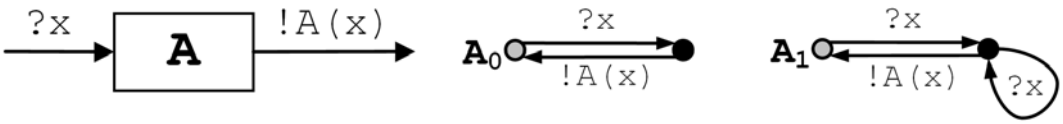


Рис.4.

Второй тип – это *дивергенция* – бесконечная последовательность внутренних действий системы, «зацикливание». При дивергенции мы также можем не получить в ответ на тестовое воздействие никакого наблюдения просто потому, что всё время будут выполняться внутренние действия. Проблеме дивергенции посвящено довольно много теоретических работ. Однако в теориях, имеющих выход в практику, то есть развитых до алгоритмов генерации тестов, обычно просто предполагается, что в реализации не должно быть дивергенции.

В то же время проблема дивергенции не в том, что она может возникнуть в реализации, а в том, что она не должна возникать при том или ином взаимодействии. К такому взаимодействию, в частности, относится взаимодействие реализации как компонента сложной, составной системы с другими, заданными компонентами. Эти другие компоненты взаимодействуют с реализацией не произвольным, а вполне определённым образом, из-за чего имеющаяся в реализации дивергенция может стать недостижимой. Есть и обратная проблема: даже при отсутствии дивергенции в компонентах она может возникнуть в их композиции как бесконечное взаимодействие этих компонентов между собой. При иерархическом построении системы такая дивергенция может, в свою очередь, оказаться промежуточной, то есть исчезающей при дальнейших композициях. Поэтому, запрещая дивергенцию на всех уровнях, мы выбрасываем из рассмотрения целый класс работоспособных систем.

Таким образом, представляется важным и полезным допустить наличие дивергенции в реализации, но при тестировании избегать тестовых воздействий в ситуации, когда возможна дивергенция.

Третий тип – это поведение, которое прямо запрещено при тестировании по тем или иным причинам. Этой теме посвящено очень мало работ. Лишь в редких случаях вводятся запрещённые состояния, в которые реализация не должна попадать. Между тем, это довольно частое явление. Например, запрещённым можно считать поведение программы при нарушении предусловия обращения к ней. Тестирование такого поведения не только бессмысленно, но может быть и опасным, если тестируемая система имеет высокий уровень доступа к привилегированным программам и данным. В частности, эта тема весьма актуальна в теории и практике информационной безопасности, где система считается безопасной, если в ней нет недекларированного поведения, которое можно понимать как запрещённое. В то же время, такая система может быть построена из компонентов, которые сами по себе не являются безопасными, но запрещённое поведение недостижимо при работе системы в целом за счёт того, что каждый компонент взаимодействует не с произвольным окружением, а с заданными компонентами системы.

В целом, является актуальной задача расмотрения в теории конформности всех трёх указанных типов ненаблюдаемого поведения. Спецификация должна позволять описывать случаи, когда в реализации допустимы (не влияют на конформность) ненаблюдаемые отказы, дивергенция и запрещённое поведение. В то же время при тестировании их следует избегать, что требует определённой модификации самой конформности. Особое внимание должно быть уделено дивергенции и запрещённому поведению в случае сложной системы, собранной из отдельных компонентов.

Асинхронное тестирование

Ещё одной проблемой является степень доступа теста к реализации. На практике чаще приходится иметь дело не с синхронным, а с асинхронным взаимодействием теста и реализации. В стандарте ISO [141] асинхронное тестирование определяется как тестирование реализации, помещённой в тестовый контекст. Чаще всего такой контекст понимают как пару

неограниченных очередей, буферизующих стимулы и реакции. Чтобы правильно оценивать результаты тестирования, мы должны понимать, что в реализации проходится не обязательно та трасса, которую мы наблюдаем, как это имело место при синхронном тестировании. Наблюдаемой трассе соответствует, вообще говоря, множество синхронных трасс, которые могла бы пройти реализация при таком наблюдении. Ошибка фиксируется, если ни одной из этих трасс нет в спецификации. Этот процесс называется сериализацией. Кроме того, ослабляются требования к допустимым тестовым воздействиям, поскольку теперь никаких блокировок нет: входная очередь всегда принимает посылаемый стимул. Дивергенцию и разрушение в реализации мы, по-прежнему, должны учитывать.

В общем случае тестовый контекст можно понимать как среду взаимодействия, которая моделируется подходящей LTS и компонуется с LTS-реализацией. Это может быть несколько входных и выходных очередей, очереди с приоритетами, стеки, порядок передачи может нарушаться, стимулы и реакции могут пропадать или генерироваться лишние и так далее. Для каждой такой среды определение допустимых тестовых воздействий и сериализацию нужно делать по своим правилам. Составление правил такого тестирования – это интеллектуальный труд, далеко не всегда простой.

Решением является общий метод, когда тесты генерируются не по исходной спецификации с учётом этих правил, а по композиции спецификации со средой, и по ней же проверяются наблюдения. Однако здесь возникает, так называемая, проблема несохранения конформности. Дело в том, что асинхронные тесты могут ловить ложные ошибки, то есть ошибки, не обнаруживаемые при более полном синхронном тестировании. Продемонстрируем эту проблему на двух примерах.

Первый пример – это система на Рис.4, рассматриваемая в ***ioco***-семантике (семантике с ненаблюдаемыми блокировками). При асинхронном тестировании через неограниченную входную очередь всегда можно послать два стимула `x` подряд: очередь их примет, а реализация потом будет выбирать стимулы из

очереди. Если после этого принимать реакции, то, согласно спецификации A_0, должны быть последовательно приняты две реакции A(x). В то же время, хотя реализация A_1 конформна, при асинхронном тестировании в ней будет обнаружена ошибка, поскольку второй реакции может не быть (стационарность в начальном состоянии).

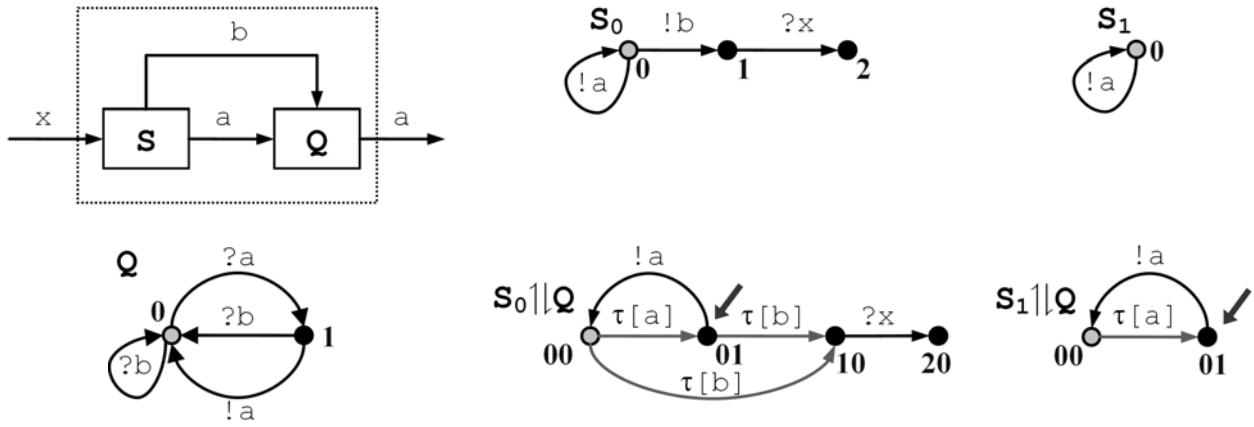


Рис.5.

Второй пример – асинхронное тестирование системы S с помощью среды взаимодействия Q (Рис.5). Среда – это одна ограниченная выходная очередь (на рисунке длины 1) с дополнительной командой «обнуления» (b). Спецификация S_0 рассматривается в семантике с наблюдаемыми блокировками (ненаблюдаемых отказов нет). Она описывает следующие требования к системе: сначала стимул блокируется, но можно выдавать цепочку реакций a, потом можно обнулить очередь (b), принять стимул x и закончить в терминальном состоянии. Реализация S_1 конформна: она не обнуляет очередь и, соответственно, не принимает стимул. Когда с очередью Q компонуется спецификация, первый посылаемый стимул не блокируется. Однако при композиции реализации S_1 такая блокировка появляется, что при асинхронном тестировании будет квалифицировано как ошибка (показано стрелками).

Верификация декомпозиции системных требований

Последние годы большое внимание уделяется тестированию составных систем, которые компонуются инкрементально и иерархически из компонентов разных уровней. Основная проблема композиции – это проблема соотношения спецификации системы и спецификаций её компонентов. Это соотношение корректно, если композиция компонентов, конформных своим спецификациям, всегда конформна спецификации системы. К сожалению, это не всегда так. Проверку правильности такого соотношения называют верификацией декомпозиции системных требований.

Конечно, если спецификация системы никак не связана со спецификациями компонентов, то трудно ожидать их правильного соотношения. Казалось бы, спецификацию системы можно получить как композицию спецификаций компонентов, выполняя её по тем же правилам, по которым сама система компонуется из реализаций компонентов. Но оказывается, что полученная таким способом спецификация системы может быть некорректной. Эта проблема называется проблемой сохранения конформности при композиции [66] или *проблемой монотонности* [105]. В диссертации проблема несохранения конформности при асинхронном тестировании рассматривается как особый случай общей проблемы монотонности [66].

Примером может служить система, составленная из двух компонентов: передатчика и приёмника на Рис.2. При несогласованности спецификаций компонентов может возникнуть тупик во взаимодействии: передатчик начинает с передачи сообщения через один порт, а приёмник начинает приём через другой порт (Рис.6). Если спецификация системы в целом не предусматривает такого тупика, то это означает, что спецификации компонентов с ней не согласованы.

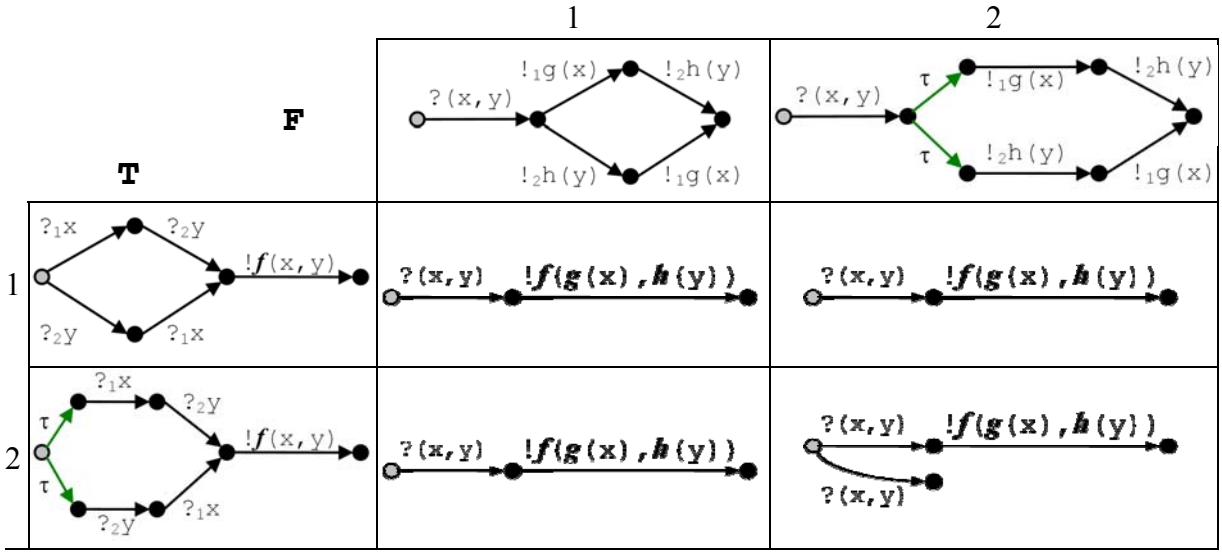


Рис.6.

Для согласования требуется либо выдача передатчиком сообщений сразу по двум портам (после передачи сообщения через один порт оставшееся сообщение передаётся через другой порт) – строка 1, либо приём приёмником сообщений сразу по двум портам (после приёма сообщения через один порт принимается оставшееся сообщение через другой порт) – столбец 1, либо и то и другое.

Отсутствие решения проблемы монотонности препятствует верификации декомпозиции системных требований. Наоборот, если найти такое решение, то можно не только верифицировать согласованность имеющейся спецификации системы со спецификациями компонентов, но и автоматически получить корректную спецификацию системы по заданным спецификациям компонентов. Сгенерированную спецификацию системы можно использовать как описание системы для её пользователей, как «техническое задание» разработчику системы при дальнейших модификациях системы и её компонентов, когда предполагается сохранение внешней функциональности системы (регрессионное тестирование), и, наконец, для генерации системных тестов.

Цели и задачи исследования

Целью диссертационной работы является создание теории конформности, которая обладала бы следующими свойствами:

1. могла применяться к широкому классу семантик взаимодействия, основанных на наблюдаемом поведении, с параметризацией конформности типа редукции той или иной семантикой из этого класса;
2. учитывала ненаблюдаемые отказы, дивергенцию и запрещённое поведение, допустимые в реализации, а не просто запрещала их всегда;
3. была развита до уровня алгоритмов генерации тестов;
4. решала проблемы сохранения конформности при асинхронном тестировании и верификации декомпозиции системных требований.

Достижение этой цели требует решения следующих задач.

1. Формализация тестового эксперимента с целью определения семантики взаимодействия, основанной на наблюдаемом поведении и параметризуемой тестовыми возможностями по управлению и наблюдению. Элементы такой семантики описывают управление и наблюдение и состоят из: 1) *действий*, которые может выполнять реализация и которые наблюдаемы тестом, 2) *кнопок*, на которых «написаны» множества разрешаемых каждой кнопкой действий (элемент управления), 3) *наблюдаемых отказов* – для тех «кнопочных» множеств, для которых возможно наблюдение отсутствия действий. Параметризация должна обеспечиваться возможностью задания любых (но согласованных между собой) наборов таких элементов.

2. Построение трассовой теории, в которой модель – это множество трасс наблюдений (действий и отказов) с ограничениями, вытекающими из семантики взаимодействия. С целью внесения в теорию дивергенции и запрещённого поведения следует предусмотреть в спецификации возможность завершать такие трассы специальными действиями, моделирующими дивергенцию и запрещённое поведение (действие, названное *разрушением*).

Определение в рамках трассовой теории для заданной семантики взаимодействия следующих понятий: 1) безопасное тестирование как

тестирование, избегающее ненаблюдаемых отказов, дивергенции и разрушения в реализации, 2) гипотеза о безопасности реализации, определяющая класс безопасных реализаций, то есть реализаций, которые могут безопасно тестироваться для данной спецификации, 3) безопасная конформность: конформность типа редукции, определяющая класс конформных реализаций для заданной спецификации и проверяемая при безопасном тестировании.

Разработка методов генерации наборов тестов по заданной трассовой спецификации в заданной семантике взаимодействия, обеспечивающих полноту тестирования. Предлагаемые методы должны поддаваться алгоритмизации при достаточно слабых ограничениях на спецификацию (то есть ограничениях, обычно выполняемых в практических системах).

3. Построение LTS-теории, включающей определение LTS-модели с разрушением, трасс наблюдений LTS-модели, гипотезы о безопасности, безопасной конформности, а также включающей (алгоритмизуемые) методы генерации тестов. Кроме того, определяется композиция LTS-моделей, которую невозможно определить в теории трасс наблюдений.

4. Разработка (алгоритмизуемых) методов пополнения спецификаций, целью которого является переход к такой семантике взаимодействия, в которой наблюдаемы все отказы (для всех кнопок). При таком переходе должен сохраняться класс конформных реализаций и не сужаться класс безопасных реализаций. Этим, во-первых, достигается самоприменимость спецификации (она конформна сама себе), то есть рефлексивность безопасной конформности. При наличии ненаблюдаемых отказов конформность, вообще говоря, нерефлексивна; из-за этого спецификация не может рассматриваться как одна из конформных реализаций, что прямо противоречит интуиции разработчика. Во-вторых, пополнение спецификаций является первым шагом к решению проблемы монотонности.

5. Построение теории верификации композиции с целью решения проблемы монотонности (сохранения конформности при композиции и при асинхронном тестировании). С учётом решения проблемы пополнения

спецификаций эту теорию достаточно построить только для семантик, в которых все отказы наблюдаемы. Проблема монотонности решается с помощью монотонного преобразования спецификации, то есть такого, что композиция преобразованных спецификаций компонентов оказывается корректной спецификацией системы (ей конформны композиции любых конформных реализаций компонентов). Такие преобразованные спецификации названы монотонными. Требуется, чтобы монотонное преобразование сохраняло классы конформных и безопасных реализаций.

Кроме того, композиция монотонных спецификаций компонентов системы, рассматриваемая как реализация, должна быть конформна заданной спецификации системы. Тогда, по транзитивности конформности, любая корректная спецификация системы предъявляет к ней не больше требований, чем композиция монотонных спецификаций компонентов. Тем самым, задача верификации декомпозиции системных требований сводится к проверке того, что композиция монотонных спецификаций компонентов конформна заданной спецификации системы. В теории верификации должно быть не только доказано существование монотонных преобразований, но и построено такое преобразование, которое, во-первых, по возможности не слишком увеличивает размер спецификации и, во-вторых, поддаётся алгоритмизации. Кроме этого, монотонное преобразование должно быть применимо в асинхронном тестировании, когда преобразованию подлежит только реализация, но не фиксированная и известная среда взаимодействия.

Методологическая и теоретическая основа исследования

Для определения семантики взаимодействия и конформности в диссертации использовалось математическое моделирование тестового эксперимента, что позволило осуществить дедуктивный анализ математических моделей реализации, спецификации и теста. При построении и анализе этих математических моделей, при разработке методов генерации тестов и различных преобразованиях моделей, использовались элементы теории

автоматов и формальных языков, теории графов и теории алгоритмов [28-37,45-48,51,52,55,81 и др.], а также формализм систем помеченных переходов (LTS) [141,99-101,104,117,118,122 и др.]. Для построения теории верификации композиции применён специально разработанный автором метод моделирования с помощью, так называемых, ϕ-трасс, позволивший объединить в рамках одной теории трассовый подход, достаточный для определения безопасности и конформности, и композицию моделей, которая обычно определяется только для LTS.

Научная новизна исследования

Новым в диссертационной работе являются:

1. Метод формализации тестового эксперимента с помощью параметризуемой машины тестирования, целью которого является определение семантики взаимодействия.
   a. Отличие от машин Милнера и Ван Глаббека заключается в параметризации машины набором «множественных» кнопок. Это является способом задания семантики взаимодействия, основанной только на наблюдениях (не на соответствии состояний) и моделирующей тестовые возможности по управлению и наблюдению. Впервые появляется возможность единообразно определять не только семантики, задаваемые машинами Милнера и Ван Глаббека, но и многие другие семантики, применяемые в теории и на практике, в частности, семантику отношения ***ioco***.
   b. Вводится новое понятие разрушения, означающее специальное действие, которое моделирует запрещённое поведение системы. В отличие от запрещённого состояния, разрушение, как запрещённое действие, может быть введено на уровне машины тестирования и в дальнейшем использовано в трассовой модели, а не только в модели LTS.
2. Введение на уровне трассовой и LTS-моделей новых взаимосвязанных концепций:

a. Безопасное тестирование как тестирование, избегающее ненаблюдаемых отказов и разрушения и не продолжающее тестовый эксперимент при дивергенции. Это является хорошей альтернативой полному запрету на поведение таких типов, что позволяет расширить домен конформности.
b. Гипотеза о безопасности – гипотеза о реализации, которая позволяет безопасно тестировать реализацию для заданной спецификации.
c. Безопасная конформность ***saco*** (*safe conformance*), которая опирается на гипотезу о безопасности, параметризуется семантикой взаимодействия и проверяема при безопасном тестировании. Многие частные конформности, основанные на семантиках наблюдаемого поведения, такие как отношение ***ioco***, являются частными случаями отношения ***saco***.

3. Метод генерации полного набора тестов для безопасной конформности по заданной спецификации, который является обобщением аналогичных методов для частных конформностей (в том числе, отношения ***ioco***) и алгоритмизуем при достаточно слабых ограничениях на спецификацию (то есть ограничениях, обычно выполняемых в практических системах). В отличие от аналогичных методов спецификация может быть задана не только в форме LTS, но и в форме модели наблюдаемых трасс. Кроме того, обеспечивается безопасное тестирование даже в тех случаях, когда в реализации есть ненаблюдаемые отказы, дивергенция и разрушение при условии, что реализация удовлетворяет гипотезе о безопасности.
4. Метод пополнения спецификаций, сохраняющего класс конформных и не сужающего класс безопасных реализаций, с использованием специальных фиктивных действий «не-отказов». Этот метод позволяет перейти к семантике, в которой все отказы наблюдаемы и которая является отношением предпорядка (в частности, рефлексивна). Метод поддаётся алгоритмизации при достаточно слабых ограничениях на спецификацию. Частное применение этого метода решает проблему пополнения для отношения ***ioco***.

5. Введение ϕ-трасс и соответствующей модели ϕ-трасс как промежуточного уровня абстракции между моделью наблюдаемых трасс, достаточной для определения конформности, но не достаточной для определения композиции моделей, и уровнем LTS-моделей, где такая композиция определяется. Эта модель позволяет в единой трассовой теории формализовать как конформность, так и композицию, что является ключевым моментом в проблеме монотонности. В частности, по ϕ-модели восстанавливается модель наблюдаемых трасс, а композиция ϕ-моделей обладает важным свойством аддитивности: ϕ-модель, соответствующая композиции LTS-моделей, совпадает с композицией ϕ-моделей, соответствующих LTS-операндам.
6. Метод монотонного преобразования спецификаций для семантики, в которой все отказы наблюдаемы. Этот метод позволяет сохранить при преобразовании классы конформных и безопасных реализаций и применим в случае асинхронного тестирования. Метод поддаётся алгоритмизации при достаточно слабых ограничениях на спецификацию. Частное применение этого метода решает проблему монотонности для семантики отношения ***ioco***.

Практическая значимость работы

Хотя диссертационная работа носит теоретический характер, она имеет практическую ценность, прежде всего, как теоретическая основа существующих и разрабатываемых методов спецификации, генерации тестов и верификации декомпозиции. Параметризация конформности позволяет единообразно использовать одни и те же понятия и методы в зависимости от тех или иных тестовых возможностей по управлению тестируемой системой и наблюдению её поведения. Предлагаемая теория позволяет яснее понимать, какие слабые места имеются у практических тестовых инструментов и какого рода ошибки могут оставаться в программах после тестирования этими инструментами. Также она формулирует те ограничения, которым должны удовлетворять спецификации, чтобы по ним можно было генерировать полный

набор тестов или компоновать спецификацию составной системы по спецификациям компонентов. Наконец, теория выявляет те гипотезы о реализации, которые обычно подразумеваются, но редко формулируются точно и недвусмысленно, что иногда приводит к недоразумениям при тестировании или верификации композиции.

В диссертационной работе важное значение придавалось поиску тех, по возможности, как можно более слабых ограничений, которые нужно налагать на исходные спецификации, чтобы их можно было алгоритмически задавать, автоматически генерировать по ним полные наборы тестов и выполнять пополнение спецификаций и их монотонное преобразование, а также алгоритмически компоновать преобразованные спецификации.

Основные практические результаты работы – это предлагаемые методы генерации тестов и преобразования спецификаций:

1. Общий метод генерации тестов позволяет создавать тестовые системы по единой схеме для разных семантик тестового взаимодействия.
2. Появляется возможность единообразного асинхронного тестирования с разными средами взаимодействия на базе предлагаемого преобразования спецификации и последующей её композиции со средой.
3. Метод композиция преобразованных спецификаций компонентов составной системы является основой для создания технологии автоматической верификации декомпозиции системных требований. При этом важно, что спецификации компонентов и системы в целом могут пониматься в разных семантиках тестового взаимодействия.

Апробация результатов исследования

Теория конформности, предлагаемая в диссертационной работе, создавалась в результате осмысления и обобщения практического опыта тестирования на основе формальных моделей, которое проводилось в ИСП РАН, начиная с 1994 г. Многие основные идеи, понятия и методы, предложенные в работе, находили своё применение в создании и дальнейшем

развитии тестовых систем KVEST (Kernel VErification and Specification Technology) и UniTESK (**Uni**fied **TE**sting & **S**pecification **T**ool**K**it) [1-15,19-23,25-27,38,49,50,53]. В первую очередь это относится к понятиям разрушения и безопасного тестирования, а также к преобразованиям спецификации при асинхронном тестировании с различными средами взаимодействия.

Результаты диссертационной работы были доложены на следующих Российских и международных научных конференциях и семинарах:

- Всероссийская научная конференция "Научный сервис в сети интернет: технологии распределенных вычислений", Абрау-Дюрсо, 2005;
- Семинар Института проблем информационной безопасности при МГУ им. М.В. Ломоносова, 2005 г.;
- Европейская объединённая конференция по теории и практике программного обеспечения ETAPS, семинар «Тестирование на основе моделей», Вена, Австрия, 2006 г.;
- Пятая общероссийская научная конференция «Математика и безопасность информационных технологий» (МаБИТ-06) в рамках Международной научной конференции по проблемам безопасности и противодействия терроризму, Москва, 2006 г.;
- Семинар кафедры математической кибернетики ф-та ВМиК МГУ им. М.В. Ломоносова, Москва, 2006г.;
- Семинар «Тестирование дискретных управляющих систем на основе автоматных моделей», Томский Государственный Университет, 2006 г.;
- Научный семинар «Проблемы современных информационно-вычислительных систем», мех.-мат. фак-т МГУ им. М.В. Ломоносова, Москва, 2006 и 2007 г.г.

По тематике диссертационной работы автор участвовал в выполнении следующих грантов РФФИ:

- 96-01-01277-а «Методы тестирования программного обеспечения на основе формальных спецификаций» (1996-1998);

- 99-01-00207-а «Формальные методы в форвард- и реверс-инженерии программных систем» (1999-2001);
- 02-01-00959-а «Повторное использование формальных спецификаций в производстве и тестировании программных систем» (2002-2004);
- 04-07-90308-в «Верификация функций безопасности и мобильности протоколов IP» (2004-2006);
- 05-01-00999-а «Методы формальных спецификаций в тестировании Интернет–приложений» (2005-2007);
- 07-07-00243-а «Верификация функций безопасности протокола нового поколения IPsec v2» (2007-2009).

Содержание диссертационной работы легло в основу разработанного совместно с А.С. Косачевым учебного курса «Теория верификации соответствия программ», который читался на механико-математическом ф-те МГУ им. М.В. Ломоносова весной 2006 г. Отдельные элементы теории используются в учебном курсе «Тестирование на основе моделей», который читается на ф-те ВМиК МГУ им. М.В. Ломоносова, начиная с 2007 г. и в курсе «Методы формальной спецификации программ», который читался на 3-ем потоке ф-та ВМиК МГУ им. М.В. Ломоносова в 1995-2007 г.г.

Публикации

По теме диссертационной работы опубликовано 27 работ, полностью отражающих основные научные результаты работы, в том числе 7 в рецензируемых научных изданиях по списку ВАК 2007 года.

Структура и объём работы

Диссертационная работа состоит из введения, шести глав, заключения, списка литературы и приложения, содержащего доказательства утверждений, встречающихся в тексте. Каждая глава разбита на разделы, некоторые разделы разбиты на подразделы; каждый уровень имеет независимую нумерацию

внутри единицы вышестоящего уровня. 135 определений, 84 леммы и 49 теорем имеют раздельную сквозную нумерацию по всему тексту. Содержание работы изложено на 434 страницах (без приложения), список библиографических ссылок включает 154 наименования (для удобства пользования сначала перечисляются работы автора, затем другие работы на русском языке и, наконец, работы на других языках). В тексте содержится 87 рисунков.

Глава 1. Используемые понятия и обозначения

Структура главы:

1. Классы, множества, числа и соответствия
2. Последовательности
3. Деревья последовательностей
4. Порождающий граф

Здесь вводятся встречающиеся в тексте общематематические понятия и обозначения.

1.1. Классы, множества, числа и соответствия

Мы опираемся на NBG (Нейман, Бернайс, Гедель) аксиоматическую теорию множеств с праэлементами (от немецкого Urelement) [29]. Эта теория основана на понятиях класса и праэлемента. Обычные теоретико-множественные операции и отношения (конструктор, объединение, пересечение, вложенность, декартовое произведение) определяются для класса, который считается состоящим из элементов. Такими элементами могут быть праэлементы и некоторые классы, называемые множествами. Праэлемент – это объект, который является элементом класса, но сам классом не является. Множество – это класс, который является как элементом другого класса, так и сам состоит из элементов (множеств и праэлементов). Собственный класс – это класс, который не является множеством, то есть не является элементом какого-либо класса, хотя сам состоит из элементов (множеств и праэлементов).

Определение 1:

- $\mathcal{N}$ – множество натуральных чисел;
- $\mathcal{N}_0 =_{\text{def}} \mathcal{N} \cup \{0\}$ – множество целых неотрицательных чисел.
- Введём символ бесконечности ∞, и определим $\mathcal{N}_{0\infty} =_{\text{def}} \mathcal{N}_0 \cup \{\infty\}$

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 $k\in\mathcal{N}_{0\infty}$:

  $[n..k] =_{def} \{i\in\mathcal{N} \mid n\le i\le k\}$.

- Для класса L:
  - $\mathcal{P}(L) =_{def} \{A \mid A\subseteq L\}$ – класс всех множеств, являющихся подклассами класса L,
  - $|L|$ – *мощность класса* L;
    для бесконечного класса L будем обозначать $|L|=\infty$.[2]
- *Соответствие* между двумя классами – это подкласс их декартового произведения. Для соответствий $f\subseteq A\times B$, $g\subseteq C\times A$ и элементов $a\in A$ и $b\in B$ обозначают:
  - $af b \quad =_{def} f(a,b) =_{def} (a,b)\in f$;
  - $f(a) \quad =_{def} \{b\in B \mid af b\}$ – образ элемента a;[3]
  - $\mathbf{Dom}(f) =_{def} \{a\in A \mid \exists b\in B\ af b\}$ – область определения;
  - $\mathbf{Im}(f) =_{def} \{b\in B \mid \exists a\in A\ af b\}$ – область значений;
  - $f^{-1} \quad =_{def} \{(b,a) \mid af b\}$ – обратное соответствие;
  - $f\circ g \quad =_{def} \{(c,b) \mid c\in C \ \&\ b\in B \ \&\ \exists a\in A\ cga \ \&\ af b\}$
    – композиция соответствий.
- *Отображением* называется однозначное соответствие. Мы будем рассматривать, как правило, всюду определённые отображения:

---

[2] Это обозначение не совсем корректно, но для наших целей годится: оно всего лишь фиксирует тот факт, что мощность класса не конечна, а мы не будем различать между собой мощности, отличные от конечных.

[3] Обозначение $f(a)$ для соответствий общего вида не вполне корректно, так как такое же обозначение применяется для отображений (однозначных соответствий), но там оно означает, не множество из одного элемента, а сам этот элемент. Чтобы избежать этой двусмысленности, для соответствий вместо $f(a)$ часто пишут $im_f(a)$. Но в данной работе $f(a)$ для отображений всегда будет означать элемент, а для остальных соответствий – множество.

$f: A \to B \;=_{def}\; f \subseteq A \times B \;\&\; \forall a \in A \;|\{b \in B \mid a f b\}| = 1.$

- Для отображения $f: A \to B$, элемента $a \in A$ и подкласса $C \subseteq A$:
  - $f(a) \;=_{def}\; b \in B$ такое, что $a f b$;
  - $f(C) \;=_{def}\; \{f(a) \mid a \in C\}$.
- ***Bool*** $=_{def}\; \{$***true***, ***false***$\}$.
- *Предикатом* называется отображение в ***Bool***.
- Если задано отображение $f: A \to B$, то, по умолчанию, будем считать также заданным отображение $f: \mathcal{P}(A) \to \mathcal{P}(B)$, 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. Мы будем использовать префиксную бесскобочную запись:

  $\cup K \;=_{def}\; \{x \mid \exists y \in K \; x \in y\}$,

  $\cap K \;=_{def}\; \{x \mid \forall y \in K \; x \in y\}$.
- При отсутствии скобок операции (отображения) выполняются в порядке следования их идентификаторов слева направо. Исключение составляют a) арифметические, b) логические и c) операции над классами, для которых используется обычная система приоритетов: a) возведение в степень приоритетнее умножения, умножение приоритетнее сложения, b) отрицание приоритетнее конъюнкции, конъюнкция приоритетнее дизъюнкции, c) пересечение приоритетнее объединения.

□

1.2. Последовательности

Определение 2:

- Последовательностью длины $n \in \mathcal{N}_{0\infty}$ в алфавите (множестве) L называют инъекцию $\sigma : [1..n] \to L$.
  - $|\sigma| = |Dom(\sigma)| = n$ – длина последовательности.
  - ϵ – пустая последовательность (нулевой длины).
- Множества последовательностей в алфавите L:
  - L^{∞} – бесконечные последовательности;
  - L^{*} – конечные ($n \neq \infty$) последовательности;

    будем считать, что всегда (в том числе для пустого L) $\epsilon \in L^{*}$.
  - $L^{\omega} =_{def} L^{*} \cup L^{\infty}$ – 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 $f : Dom(f) \to L$ 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):

  $$\langle f(i) \,|\, p(i) \rangle =_{def} \{(i, f(j)) \;|\; i \in [1..|I|] \;\&\; j = \nu_I(i)\},$$

  где $I = p^{-1}(true) \cap Dom(f)$.
- Для $z_1, \ldots z_n \in L$: $\langle z_1, \ldots z_n \rangle =_{def} \sigma$ такая, что

$\boldsymbol{Dom}(\sigma)=[1..n]$ и $\forall i\in\boldsymbol{Dom}(\sigma)\ \ \sigma(i)=z_i$.

- Для $\sigma\in L^{\omega}$ последовательность $\langle\sigma(i)\,|\,\boldsymbol{p}(i)\rangle$ называется *подпоследовательностью*.
- Для последовательности $\sigma\in L^{\omega}$ и класса 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 $n\in\boldsymbol{\mathcal{N}}_0$ и $k\in\boldsymbol{\mathcal{N}}_{0\infty}$.

  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 $\cdot\cdot: L^{*}\times L^{\omega}\to L^{\omega}$.

  Для $\mu\in L^{*}$, $\lambda\in L^{\omega}$: $\mu\cdot\lambda =_{def} \sigma$ 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

  $\cdot\cdot:\mathcal{P}(L^{*})\times\mathcal{P}(L^{\omega})\to\mathcal{P}(L^{\omega})$:

  для $\mathbf{M}\subseteq L^{*}$, $\mathbf{\Lambda}\subseteq L^{\omega}$ $\mathbf{M}\cdot\mathbf{\Lambda} =_{def} \{\mu\cdot\lambda\,|\,\mu\in\mathbf{M}\ \&\ \lambda\in\mathbf{\Lambda}\}$;

· Для бинарного оператора $.\otimes.:L^*\times L^*\to L^*$ естественным образом определяется унарный оператор, применяемый к последовательности (конечных последовательностей, множеств конечных последовательностей и т.д.) $\otimes.:L^{*\omega}\to L^{\omega}$, $\otimes.:\mathcal{P}(L^*)^{\omega}\to\mathcal{P}(L^{\omega})$ и т.д.

$\otimes(\langle z_1,z_2,z_3,...\rangle =_{def} z_1\otimes z_2\otimes z_3\otimes...$ с естественным порядком выполнения слева направо.

При этом считается, что для одноэлементной последовательности $\oplus(\langle\sigma\rangle)=\sigma$, где $\sigma\in L^*$, а для пустой последовательности $\oplus(\epsilon)=\epsilon$.

Таким оператором является, в частности, конкатенация $\cdot$.

· Для $\mu,\sigma\in L^{\omega}$: $\mu\le\sigma =_{def} \mu=\sigma[1..|\mu|]$, 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.

□

1.3. Деревья последовательностей

Определение 3:

· *Деревом последовательностей* в алфавите L назовём такое множество последовательностей $\Sigma\subseteq L^{\omega}$, которое вместе с каждой последовательностью содержит любой её префикс: $\forall\sigma\in\Sigma\ \forall\mu\le\sigma\ \mu\in\Sigma$. Такое множество последовательностей называют ещё замкнутым по взятию префикса или префикс-замкнутым (prefix-closed [90]).[4]

· Дерево конечно, если оно конечно как множество (последовательностей).

---

[4] 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.

- Дерево последовательностей будем называть *нётеровым*, если в нём нет бесконечно возрастающей цепочки последовательностей[5].
- Множество всех деревьев последовательностей в алфавите L обозначим $\boldsymbol{Trees}^{\omega}(L)$, множество всех деревьев *конечных* последовательностей в алфавите L обозначим $\boldsymbol{Trees}(L)$.
- Определим операцию взятия префикса последовательности: $\mu\cdot\lambda\rightarrow\mu$. Замыкание по этой операции, то есть множество префиксов последовательности $\sigma\in L^{\omega}$, очевидно, является деревом, включающим исходную последовательность, и мы будем обозначать это дерево

  $\boldsymbol{Tree}(\sigma) =_{def} \{\mu\in L^{\omega} \mid \mu\le\sigma\}$.

  Соответственно, пополнение произвольного множества последовательностей $\mathbf{T}\subseteq L^{*}$ префиксами всех его последовательностей также является деревом $\cup_{\circ}\boldsymbol{Tree}(\mathbf{T})$.
- Для дерева $\boldsymbol{\Sigma}\in\boldsymbol{Trees}^{\omega}(L)$:
  - $\boldsymbol{\Sigma}$ *ограниченное*, если множество длин его трасс ограничено сверху конечным числом: $\exists N\in\mathcal{N}\ \forall\sigma\in\boldsymbol{\Sigma}\ |\sigma|<N$;
  - множество максимальных (по отношению $\le$) последовательностей дерева обозначим

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

    $\boldsymbol{head}(\boldsymbol{\Sigma}) =_{def} \{z\in L \mid \langle z\rangle\in\boldsymbol{\Sigma}\}$;
  - для $\sigma\in L^{*}$: $\boldsymbol{\Sigma}\ \boldsymbol{after}\ \sigma =_{def} \{\lambda\in L^{\omega} \mid \sigma\cdot\lambda\in\boldsymbol{\Sigma}\}$;

    очевидно, $\boldsymbol{\Sigma}\ \boldsymbol{after}\ \sigma\in\boldsymbol{Trees}^{\omega}(L)$;

---

[5] В частности, в нётеровом дереве нет бесконечной последовательности, хотя отсутствие бесконечных последовательностей ещё не гарантирует нётеровости.

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

□

Определение 4: Индуктивный предел. *Пределом* бесконечно возрастающей цепочки конечных последовательностей будем называть бесконечную последовательность, каждый префикс которой является префиксом некоторой последовательности в цепочке. Будем говорить, что множество (конечных и бесконечных) последовательностей *замкнуто по пределу* или *предельно*, если оно содержит пределы всех своих бесконечно возрастающих цепочек последовательностей.

□

1.4. Порождающий граф

Для наглядности мы будем использовать в примерах представление множества последовательностей порождающим графом.

Определение 5: Порождающим графом называется ориентированный граф, дуги которого могут быть помечены символами алфавита. Допускаются пустые (непомеченные) 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.

□

В большинстве случаев нас будут интересовать только конечные последовательности, порождаемые графом. По контексту всегда будет ясно, что означает выражение «множество последовательностей, порождаемых графом»: множество всех таких последовательностей или только подмножество конечных последовательностей.

В примерах мы будем помечать начальную вершину порождающего графа серым кружком (см. Рис.7).

Пример порождающего графа

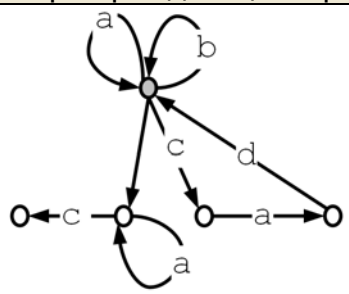


Рис.7.

Теорема 1: Необходимым и достаточным условием того, что множество последовательностей – это дерево, является существование порождающего это множество графа, в котором все вершины конечные и одна начальная.

[6]□435

Множество последовательностей *регулярно*, если оно может быть задано конечным порождающим графом [32,51].

[6] Здесь и далее в формулировке Лемм и Теорем после знака «□» указывается номер страницы в Приложении, где начинается доказательство утверждения. Исключение составляет несколько случаев ссылки на статьи, в которых эти утверждения доказаны; в этих случаях указывается номер ссылки на статьи по списку литературы в форме «см.[номер]».

Глава 2. Формализация тестового эксперимента

Структура главы:

1. Спецификационная и реализационная модели. Конформность
2. Тестирование конформности
3. Реализационная модель
4. Управление и наблюдение
5. Остановка машины и наблюдение отказов
6. Взаимодействие теста и реализации. Безопасное тестирование
7. Гипотеза о безопасности и безопасная конформность
8. Пополнение спецификаций
9. Монотонность конформности
10. Разрушение
11. Дивергенция
12. Примеры тестовых семантик и конформност
13. Проблема выбора семантики тестового взаимодействи
14. Трассы готовности
15. Репликация и симуляции
16. Глобальное тестирование
17. Бесконечные и отрицательные наблюдения
18. Приоритеты
19. Выводы

2.1. Спецификационная и реализационная модели. Конформность

Правильность исследуемой системы в самом широком смысле понимается как её соответствие заданным требованиям. Такое соответствие будем называть реальной конформностью. Для верификации (проверки) конформности с помощью формальных методов, объекты и отношения реального мира

отображаются в модельные, математические объекты и отношения. Модель исследуемой системы мы называем реализацией (сокращённо от «реализационная модель»[7]), модель требований – спецификационной моделью или, сокращённо, спецификацией, а реальная конформность отображается в модельную конформность. Последняя понимается как обычное математическое соответствие, то есть подмножество декартового произведения множеств реализаций и спецификаций (Рис.8) [141].

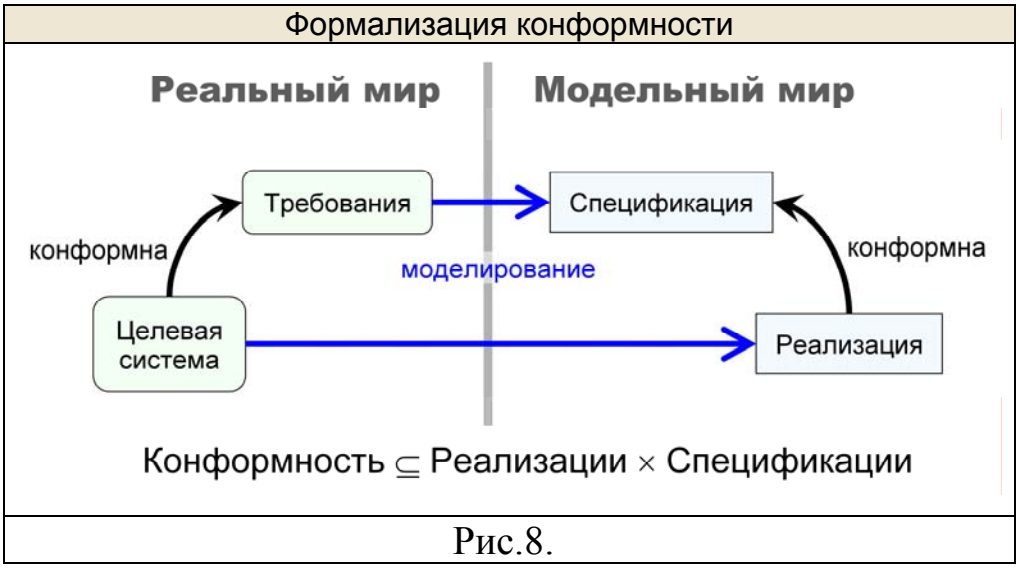


Рис.8.

Верификация конформности основана на предположении, что моделирование выполнено «правильно». Это даёт возможность считать реальную и модельную конформности эквивалентными: система конформна требованиям тогда и только тогда, когда реализация, как модель системы, конформна спецификации, как модели требований. Конечно, это утверждение тавтологично в том смысле, что, если бы мы захотели дать формальное определение «правильности» моделирования, то оно свелось бы к эквивалентности реальной и модельной конформностей. Тем самым, верификация конформности проверяла бы ровно то, что она должна

[7] Слово «реализация» также соответствует английскому термину “*implementation*”, традиционно используемому в англоязычной литературе по тестированию в значении «модель тестируемой системы».

предполагать. Чтобы выйти из этого порочного круга, нужно понять саму природу моделирования.

Моделирование требований – это процесс их уточнения и формализации [38,43]. Хотя результатом становится формальное описание требований (спецификация), сам процесс вряд ли может быть формализован, поскольку связывает объекты разной природы: неформальный и формальный. Только наша интуиция может подсказать, правильно ли мы записали неформальные требования в виде формальной спецификации. Если интуиция нас подвела, это может привести (и часто приводит) к выявлению ложных ошибок в исследуемой системе. Тогда приходится пересматривать моделирование требований, то есть выявлять ошибку не в системе, а в спецификации. После исправления спецификации верификацию конформности приходится повторять.

Моделирование конформности имеет ту же природу: это уточнение и формализация нашего интуитивного представления о том, что означает утверждение «система удовлетворяет требованиям». Формализация предполагает также, что мы абстрагируемся от тех деталей системы, требований и их отношения, которые несущественны с точки зрения интуитивно понимаемой «правильности» системы.

Для верификации конформности нам должны быть заданы в той или иной, но формальной, форме спецификация и конформность. Если реализация, как модель исследуемой системы, также известна, то возможна статическая, аналитическая верификация. Такая верификация сводится к проверке того, что пара <реализация, спецификация> принадлежит допустимому множеству таких пар, определяемому модельной конформностью.

В отличие от требований и реальной конформности, исследуемая система – это обычно программный и/или аппаратный комплекс, который на известном уровне абстракции может быть понят вполне формально. Тем самым, отображение системы в реализацию связывает два формальных объекта, что даёт потенциальную возможность определить (и выполнять) это отображение

формально. Иногда такое отображение действительно возможно и применяется на практике. Проблема, однако, в том, что, как правило, система и её модель (реализация) задаются на слишком различных уровнях абстракции, что для сложной системы делает практически невыполнимым её анализ и формальное преобразование в реализацию. При моделировании мы абстрагируемся от большинства деталей устройства и поведения системы, выделяя существенное и несущественное с точки зрения заданных спецификации и модельной конформности. Выполняя такое моделирование не- или полу-формально, мы получаем ещё один источник ложных ошибок. Поэтому часто приходится вообще отказаться от анализа системы и признать, что реализация нам неизвестна, а в лучшем случае известен лишь *класс возможных реализаций*. Кроме того, на практике часто встречаются ситуации, когда исследуемая система вообще не может быть подвергнута формальному анализу. Например, удалённая система, доступ к которой возможен только по каналу связи, или программа, исходный текст которой на языке программирования утерян (формально можно было бы анализировать машинный код, но практически это почти всегда невозможно), или аппаратура как «чёрный ящик».

2.2. Тестирование конформности

Здесь возникает вопрос: как проверять конформность, если реализация неизвестна? Это оказывается возможным тогда, когда требования к системе выражены в терминах взаимодействия системы с её окружением. Становится возможным тестирование (динамическая верификация) как проверка конформности в эксперименте. Тест подменяет собой окружение и, взаимодействуя с системой, наблюдает её поведение в этом взаимодействии. Из практических соображений тест должен заканчиваться за конечное время, что вовсе не обязательно для произвольного окружения. Поэтому для того, чтобы сымитировать любое поведение окружения, используется набор тестов, который потенциально может быть бесконечным. Конечно, на практике применяются только конечные наборы тестов, но такое тестирование будет

полным (правильно определяющим конформность или неконформность) только при определённых ограничениях на тестируемую систему. Такие ограничения, выраженные в модельной форме, уточняют класс возможных реализаций и называются *реализационными гипотезами* или *моделью ошибок*, если они сводятся к утверждению о том, какие ошибки (типы неконформности) в системе бывают, а какие нет.

Естественно, возникает задача формализовать процесс тестирования. Для этого мы должны, прежде всего, предполагать, что модель системы (реализация) существует, даже если она неизвестна. Это, так называемая, *тестовая гипотеза* [64]. Далее мы должны формализовать само взаимодействие и моделировать реальные тесты модельными тестами. В модельном мире определяется формальное отношение «реализация *проходит* набор (модельных) тестов». После этого набор модельных тестов объявляется *полным* для заданной спецификации, если он удовлетворяет условию: реализация конформна спецификации тогда и только тогда, когда реализация проходит этот набор тестов. Тем самым, полный набор тестов является «вторым эталоном» или второй (после спецификации) формой описания функциональности исследуемой системы [43]. Что мы должны сделать в модельном мире? Во-первых, *доказать* существование полного набора тестов для каждой спецификации (по крайней мере, из заданного класса спецификаций) и известного класса возможных реализаций, и, во-вторых, найти способ *генерации* тестов (Рис.9).

Для практического применения нам нужно определить процедуру *трансляции* модельного теста в реальный тест. Здесь ситуация обратная моделированию системы или требований: мы не определяем модель реального объекта, а строим реальный объект по его заданной модели. После этого полный набор реальных тестов, то есть полученных трансляцией полного набора модельных тестов, прогоняется на исследуемой системе. В реальном мире также определяется отношение «система проходит набор (реальных) тестов». Наконец, делается заключение, что система конформна требованиям

тогда и только тогда, когда она проходит полный набор реальных тестов. Разумеется, этот вывод основан на предположении, что не только моделирование требований и конформности, но также трансляция тестов и соответствие реального и модельного отношений *проходит* «правильные».

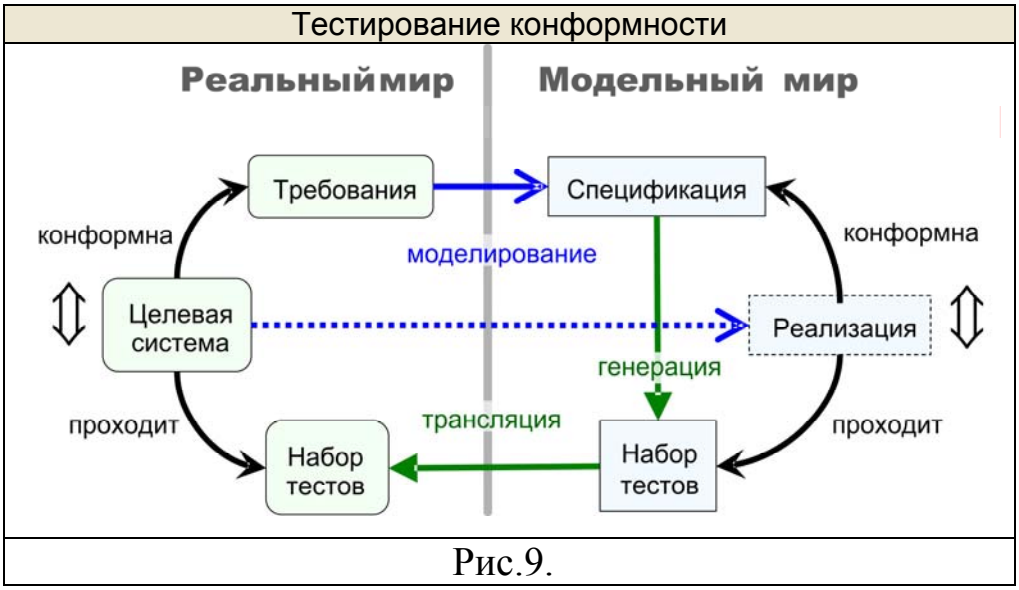


Рис.9.

Далее мы не будем останавливаться на вопросе о «правильности» моделирования и трансляции тестов, предполагая, что они «правильные». Своё внимание мы сосредоточим на модельном мире, разумеется, не забывая о мире реальном, который служит источником всех тех практических ограничений, которые нам придётся учитывать.

Формализация взаимодействия – это ключевой момент в тестировании конформности. От того, какой вид взаимодействия мы рассматриваем, непосредственно зависит конформность и допустимые классы реализаций и спецификаций. Мы будем задавать семантику взаимодействия с помощью так называемой *машины тестирования*. Взаимодействие реализации с окружением всегда подчиняется некоторым ограничениям. Нас интересует не любое взаимодействие, а только такое, которое допускается на практике Это определяется, во-первых, той операционной обстановкой в которой взаимодействие происходит, и, во-вторых, требованиями к самому окружению. Первое означает, что окружение *не может*, а второе – что окружение *не*

должно взаимодействовать с реализацией произвольным образом. Иными словами, какого-то взаимодействия не бывает, а какое-то нас не интересует, поскольку мы проверяем правильность реализации, а не её окружения. Абстрагируясь от деталей операционной обстановки и требований к окружению, мы описываем ограничения на взаимодействие с помощью подходящей машины тестирования. Такая машина определяет тот уровень абстракции и те ограничения на взаимодействия, при которых формулируется конформность в терминах такого взаимодействия. С точки зрения тестирования машина задаёт имеющиеся в нашем распоряжении тестовые возможности по управлению взаимодействием и наблюдению поведения реализации. Иными словами, интересующие нас конформности – это те отношения, для проверки которых необходимо и достаточно тестовых возможностей, описываемых машиной тестирования.

Анализ тестовых возможностей мы будем вести по трём аспектам: 1) что мы *хотим* проверять при тестировании; 2) что мы *теоретически можем* проверять с помощью данных тестовых возможностей («мощность тестирования»); 3) на какие тестовые возможности и при каких условиях мы *практически можем* рассчитывать.

Остановимся немного на первом аспекте, непосредственно определяющем тип рассматриваемой конформности при заданных тестовых возможностях. Речь идёт о том, какие поведения реализации в той или иной ситуации спецификация считает правильными, а какие нет. Мы будем предполагать, что реализация отвечает принципу независимости: любое поведение реализации в данной ситуации правильно или неправильно *независимо* от других её поведений. Тогда можно считать, что спецификация определяет в каждой ситуации t множество *разрешаемых* поведений $\mathbf{\Sigma}_{\mathrm{p}}(t)$. Если реализация демонстрирует множество поведений $\mathbf{I}(t)$, то конформность является предпорядком (рефлексивное и транзитивное отношение) и означает вложенность $\mathbf{I}(t)\subseteq\mathbf{\Sigma}_{\mathrm{p}}(t)$. Для каждого наблюдаемого поведения реализации $i\in\mathbf{I}(t)$ нужно проверить, что $i\in\mathbf{\Sigma}_{\mathrm{p}}(t)$. Такое поведение наблюдается при

однократном прогоне теста, поэтому можно при завершении прогона теста выносить вердикт ***pass*** (проходит) или ***fail*** (не проходит). Реализация проходит тест, если при любом его прогоне выносится вердикт ***pass*** (не бывает вердикта ***fail***). Реализация проходит набор тестов, если она проходит каждый тест из набора. Конформность такого рода понимается как *редукция* (сводимость) реализации к спецификации: спецификация предоставляет на выбор несколько вариантов правильного поведения, а конформная реализация может демонстрировать лишь некоторые из них [71].

Таким образом, мы не рассматриваем конформности, для проверки которых недостаточно анализировать каждое отдельное поведение реализации и требуется (дополнительный) анализ множества наблюдаемых поведений. В частности, мы не рассматриваем конформности, требующие обязательного наличия в реализации некоторых поведений. Если спецификация в ситуации t определяет множество *обязательных* поведений $\Sigma_r(t)$, то конформность требует обратной вложенности $\mathbf{I}(t) \supseteq \Sigma_r(t)$. Для проверки такой конформности требуется анализ всего множества наблюдаемых поведений $\mathbf{I}(t)$ и проверка $s \in \mathbf{I}(t)$ для каждого $s \in \Sigma_r(t)$. Разновидностью таких конформностей являются эквивалентности, когда любое разрешённое поведение обязательно должно быть в реализации, то есть $\Sigma_r(t) = \Sigma_p(t)$. Для такого рода конформностей недостаточно вердикта в конце каждого прогона теста в том смысле, что предикат конформности не является конъюнкцией вердиктов всех прогонов всех тестов из набора (если считать ***pass=true*** и ***fail=false***). Исключение составляет случай детерминированных систем, когда единственное поведение является одновременно и разрешённым и обязательным. В частности, в теории тестирования детерминированных конечных автоматов конформность понимается как эквивалентность.

Мы рассмотрим две известные машины тестирования, предложенные Милнером [121] и Ван Глаббеком [89,90], с их разнообразными модификациями. Параллельно будем определять машину с параметризованным ограниченным управлением, которая предоставляет минимальные тестовые

возможности, доступные в том или ином практическом случае. Другие тестовые возможности либо «избыточны» (позволяют проверять то, что нам не нужно), либо в настоящее время не проработаны в теории, либо «непрактичны».

В качестве примера мы будем использовать широко распространенный частный вид взаимодействия, который сводится к обмену дискретными порциями информации (сообщениями) между реализацией и окружением. Такие системы называют реактивными системами [76,117,136,149]. Сообщение, передаваемое из окружения в систему, называется *стимулом* (*input*), а сообщение, передаваемое из системы в окружение, – *реакцией* (*output*). Следуя нотации алгебры процессов CCS (Calculus of Communicating Systems [120,122]), мы обозначаем стимулы как `?m`, а реакции – как `!m`, где `m` – символ сообщения.

2.3. Реализационная модель

Поскольку наша задача – формализация взаимодействия реализации с окружением, нам безразлично, как реализация устроена «внутри». Для нас важен лишь её внешний интерфейс, позволяющий оказывать на неё тестовые воздействия и наблюдать её внешнее поведение, то есть поведение, проявляющееся во взаимодействии. В соответствии с таким подходом машину тестирования можно представлять себе как «чёрный ящик», внутри которого находится тестируемая реализация и который снабжён разного рода «устройствами» для выполнения тестовых воздействий и наблюдения внешнего поведения реализации. Эти воздействия и наблюдения совершает оператор машины, поведение которого моделирует поведение окружения. Под тестом можно понимать тот или иной вариант поведения окружения, то есть тест – это «инструкция» оператору, которую он должен выполнять. Отличие теста от такого же окружения в том, что в конце работы теста выносится вердикт ***pass*** или ***fail***.

Для наглядности и в качестве иллюстрации мы будем использовать модель исследуемой системы и её окружения, которая называется *системой помеченных переходов* – LTS (*Labelled Transition System*). Этой модели посвящена Глава 4. LTS – это ориентированный граф с выделенной начальной вершиной, дуги которого помечены некоторыми символами. Формально, LTS – это совокупность $\mathbf{S}=LTS(V_{\mathbf{S}}, L, E_{\mathbf{S}}, s_0)$, где $V_{\mathbf{S}}$ – непустое множество состояний (вершин графа), L – алфавит символов, называемых внешними действиями, τ – символ, называемый внутренним действием, $E_{\mathbf{S}} \subseteq V_{\mathbf{S}} \times (L \cup \{\tau\}) \times V_{\mathbf{S}}$ – множество переходов (помеченных дуг графа), $s_0 \in V_{\mathbf{S}}$ – начальное состояние (начальная вершина графа). Переход из состояния s в состояние s' по действию z обозначается $s \xrightarrow{z} s'$. Обозначим $s \xrightarrow{z} \not\Rightarrow =_{def} \nexists s' \; s \xrightarrow{z} s'$. Сразу отметим, что разные LTS могут быть неразличимы при тестировании, если они в любом взаимодействии демонстрируют одинаковое внешнее поведение.

Внешнее поведение рассматривается как последовательность дискретных внешних действий, выполнение которых оператор может наблюдать. Для машины задаётся алфавит L *внешних* действий. Можно считать, что внешнее действие $z \in L$ является обозначением для множества действий машины, выполнение каждого из которых наблюдаемо оператором как z, то есть для него они неразличимы между собой. Вместе с тем выполнение в одной и той же ситуации одного или другого действия, когда они оба видимы извне как z, не означает, что последствия выполнения этих действий также будут внешне неразличимы. Иными словами, внешнее поведение реализации может быть недетерминированным. Также отметим, что фиксация алфавита L не означает, что тестируемая система не может выполнять ещё какие-то действия, которые могли бы наблюдаться извне. Просто нас не интересует поведение системы при выполнении этих других действий, и поэтому они не включены в алфавит L. Например, при тестировании методов объекта, вычисляющих арифметические

функции, нас не интересует, какие ещё методы определены в классе этого объекта. В LTS-модели с алфавитом L выполнению внешнего действия $z \in L$ соответствует переход $s \text{—} z \rightarrow s`$. LTS, находясь в состоянии $s$, совершает этот переход, что наблюдается оператором как выполнение действия $z$, и оказывается в состоянии $s`$. Сами состояния s и $s`$ не наблюдаемы.

Кроме внешних, наблюдаемых действий, машина может иметь *внутреннюю активность*, которая внешне (во взаимодействии) не наблюдаема. Её обозначают символом τ. Внутренняя активность может быть как конечной, то есть она прекращается через какое-то время, так и бесконечной. Бесконечную внутреннюю активность называют *дивергенцией*. Без ограничения общности можно считать, что τ-активность представляет собой последовательность дискретных τ-действий, естественно, тоже внутренних, ненаблюдаемых. В общем, символом τ обозначается множество всех внутренних действий машины, которые не наблюдаемы оператором и тем самым для него они неразличимы между собой. Будем предполагать, что время выполнения любого внешнего или τ-действия конечно и ограничено снизу ненулевым значением. Тогда конечная последовательность действий выполняется за конечное время, а бесконечная последовательность действий выполняется бесконечно долго. Конечная τ-активность – это конечная, а дивергенция – это бесконечная последовательность выполнения τ-действий. В LTS-модели выполнению τ-действия соответствует переход $s \text{—} \tau \rightarrow s`$.

Взаимодействие всегда предполагает участие обеих сторон: реализации и окружения. Например, в реактивных системах передача сообщения означает, что окружение посылает стимул, а реализация принимает, или реализация выдаёт реакцию, а окружение её получает. Иногда говорят, что передача реакции инициируется реализацией, и, коли реакция выдана, окружение обязано её принять. В других случаях говорят, что стимул, посланный окружением, реализация не должна отвергать. Это можно рассматривать как

ограничение на допустимое поведение окружения или реализации. В общем, наличие такого рода ограничений не отменяет общего принципа взаимности взаимодействия.

В машине тестирования этот принцип означает: машина может выполнять только те действия, которые определены в реализации и разрешены оператором. В LTS-реализации действие z определено, если в текущем состоянии s есть хотя бы один переход вида s—z→s`. Тестовое воздействие сводится к тому, что оператор указывает, какие действия машине разрешено выполнять. При этом τ-действия, не являющиеся актами взаимодействия, всегда считаются разрешёнными, оператор не может их запретить. Разрешение внешних действий можно понимать и так, что оператор даёт машине приказ выполнить одно из этих действий на её выбор. Например, окружение принимает все реакции, но, какая именно реакция будет передана, определяет реализация.

Хотя выполняться может только определённое и разрешённое действие, в общем случае не всякое такое действие выполнимо. Выполнимость действия z есть предикат от множества определённых действий и множества разрешённых действий. Этот предикат может быть различным в различные моменты времени (в различных состояниях LTS-реализации) и для различных действий (для различных переходов LTS-реализации). Таким способом в машине могут быть заданы приоритеты выполнения действий. В данной работе, за исключением этой главы, мы будем рассматривать *машины без приоритетов*: всякое определённое и разрешённое действие выполнимо.

Правило недетерминированного выбора: если есть несколько выполнимых действий, то выполняется только одно из них, выбираемое недетерминированным образом. В этом кроется причина возможного недетерминизма системы, поскольку мы абстрагируемся от тех факторов («погодных условий»), которые определяют выбор того или иного действия [90]. Заметим, что даже в том случае, когда все выполнимые действия

помечены одним и тем же символом ($z \in L$ или τ), они могут быть различными, в том числе и по своим наблюдаемым последствиям. В LTS-реализации это означает, что в текущем состоянии s может быть определено несколько переходов вида $s \text{—} z \rightarrow s`$, отличающихся только конечным состоянием $s`$.

Для машины без приоритетов выбор выполняемого действия при взаимодействии моделируется в LTS с помощью оператора композиции, заимствуемого из той или иной алгебры процессов. Нам удобнее использовать оператор композиции алгебры процессов CCS [120,122]. Для этого на универсуме внешних действий определяется инволюция (биекция, обратная сама себе) «подчёркивание», которая каждому внешнему действию z ставит в соответствие *противоположное* действие $\underline{z}$ так, что $\underline{\underline{z}}=z$. Заметим, что $z \in L$ не обязательно влечёт $\underline{z} \in L$. Результатом композиции двух LTS $\mathbf{S}=LTS(V_{\mathbf{S}}, L, E_{\mathbf{S}}, s_0)$ и $\mathbf{T}=LTS(V_{\mathbf{T}}, M, E_{\mathbf{T}}, t_0)$ является третья LTS $\mathbf{S} \between \mathbf{T}=LTS(V_{\mathbf{S}} \times V_{\mathbf{T}}, L \between M, E, s_0 t_0)$. Её состояния – это пары состояний LTS-операндов, начальное состояние – пара начальных состояний. В алфавит композиции попадают те действия из алфавита каждого LTS-операнда, для которых в алфавите другого LTS-операнда не нашлось противоположных действий: $L \between M = (L \setminus \underline{M}) \cup (M \setminus \underline{L})$. Композиционные переходы делятся на асинхронные и синхронные. Асинхронному переходу соответствует переход в одном из LTS-операндов, помеченный внешним действием композиционного алфавита или символом τ; измениться может состояние только этого LTS-операнда. Синхронному переходу соответствует пара переходов в LTS-операндах по противоположным действиям, которая в композиции становится τ-переходом; измениться могут состояния обоих LTS-операндов. Формально множество композиционных переходов E – это наименьшее множество, порождаемое следующими правилами вывода:

$l \in (L \setminus \underline{M}) \cup \{\tau\}$ & $s \text{—}l\text{→} s'$ $\vdash$ $st \text{—}l\text{→} s't$;

$m \in (M \setminus \underline{L}) \cup \{\tau\}$ & $t \text{—}m\text{→} t'$ $\vdash$ $st \text{—}m\text{→} st'$;

$z \in L \cap \underline{M}$ & $s \text{—}z\text{→} s'$ & $t \text{—}\underline{z}\text{→} t'$ $\vdash$ $st \text{—}\tau\text{→} s't'$.

Если оператор в машине с LTS-реализацией **S** разрешает действие $z \in L$, то в LTS-тесте **T** этому соответствует переход вида $t \text{—}\underline{z}\text{→} t'$. Если оператор может «передумать» и, не дожидаясь выполнения внешнего действия, разрешить другое множество внешних действий, то в LTS-тесте **T** этому соответствует τ-переход вида $t \text{—}\tau\text{→} t'$. Обычно считается, что при тестировании система <реализация-тест> замкнута, то есть они взаимодействуют только друг с другом и не взаимодействуют с остальным внешним миром. В этом случае $M = \underline{L}$ и $L \between M = \varnothing$, то есть в композиции $\mathbf{S} \between \mathbf{T}$ остаются только τ-переходы (наследованные от операндов или синхронные).

Заметим, что на практике реальный тест может взаимодействовать с тестируемой системой не напрямую, а через некоторую среду взаимодействия. Стандартным примером является среда из очереди стимулов и очереди реакций в реактивных системах. Кроме того, тестируемая система может взаимодействовать не только с тестом и/или средой взаимодействия, но и некоторой другой частью окружения. Например, у нас может не быть возможности «перехватывать» всё внешнее взаимодействие системы. В этом случае в модельном мире мы должны считать, что тестируется композиция реализации со средой взаимодействия и другой частью окружения. Тогда говорят, что реализация погружена в тестовый контекст, а тестирование называют *асинхронным* или *тестированием в контексте*. Фактически, здесь меняется сама конформность. Тест также может получать команды «сверху» (изменение режима тестирования или аварийное завершение работы), что в модельном мире можно понимать как замену теста на композицию теста с

«вышестоящей инстанцией». Далее будем считать, что система <реализация-тест> замкнута.

2.4. Управление и наблюдение

Итак, в терминах машины тестирования тестовое воздействие сводится к указанию множества разрешённых внешних действий. Способ такого указания зависит от устройства машины.

Машина Ван Глаббека называется *генеративной* машиной (Рис.10): каждому внешнему действию $z \in L$ соответствует переключатель, который имеет два положения: *free* (разрешение действия) и *blocked* (запрещение действия). Можно устанавливать любые переключатели в любые положения. Множество переключателей в положении *free* – это и есть множество разрешённых действий $P \subseteq L$.

Рис.10.

Машина Милнера называется *реактивной* машиной (Рис.11) и работает «по приказу»: для каждого внешнего действия $z \in L$ в ней имеется кнопка, нажатие которой разрешает выполнять только это действие z. Это налагает ограничения на работу окружения: в LTS-модели окружения $\mathbf{T}$ в каждом состоянии должно быть не более одного перехода вида $t \xrightarrow{\underline{z}} t`$, где $z \in L \cap \underline{M}$. Отсутствие переходов означает, что в данный момент времени (в данном состоянии теста) оператор не нажимает никакой кнопки. После выполнения внешнего действия машина *приостанавливается* до нажатия следующей кнопки.

Рис.11.

Мы вводим машину тестирования *с ограниченным управлением*, которую будем называть также *параметризованной* машиной (Рис.12). Как и реактивная машина, она работает «по приказу», но кнопка разрешает не одно внешнее действие, а множество действий $P \subseteq L$; такую кнопку будем обозначать “P”. Предполагается, что каждое внешнее действие $z \in L$ разрешается хотя бы одной кнопкой, то есть существует такая кнопка “P”, что $z \in P$. Мы говорим, что наша машина – это машина с *ограниченным* управлением потому, что в ней не для каждого множества $P \subseteq L$ может существовать кнопка “P”. Машина параметризуется набором кнопок, то есть семейством подмножеств внешних действий $\mathfrak{R} \subseteq \mathcal{P}(L)$. Нажатая кнопка автоматически отжимается, если машина выполняет внешнее действие или оператор нажимает другую кнопку.

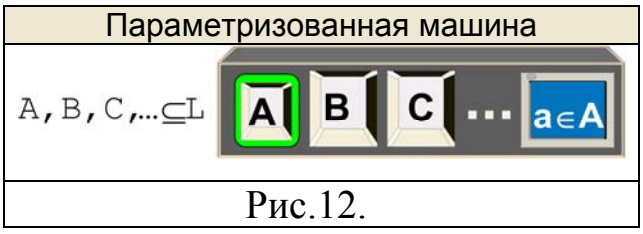


Рис.12.

Сравним эти три машины. Параметризованную машину можно понимать как модификацию генеративной машины, в которой не любое положение переключателей допустимо. Согласно Ван Глаббеку, если реактивную машину модифицировать, разрешив одновременно нажимать несколько кнопок, то такая реактивная машина будет работать аналогично генеративной машине. С учётом этих модификаций все три машины можно считать эквивалентными за одним исключением.

Таким исключением является поведение машины после выполнения внешнего действия до нового нажатия кнопок или изменения положения переключателей. Реактивная машина приостанавливается до тех пор, пока не будут ещё раз нажаты те же или другие кнопки. Генеративная машина продолжает работать, выполняя внутренние действия и внешние действия, разрешённые переключателями в положении *free*. В параметризованной машине реализован третий вариант: машина продолжает работать, но выполняться могут только внутренние действия (кнопка отжата). Посмотрим, как поведение одной машины может имитироваться в другой машине.

Ван Глаббек показал, что приостановка реактивной машины может имитироваться в генеративной машине специальным переключателем (on/off-switch), блокирующим выполнение не только всех внешних, но и внутренних действий. Для имитации оператор должен сразу после наблюдения внешнего действия нажать блокирующий переключатель, а после установки основных переключателей убрать блокировку. Заметим, однако, что оператор моделирует окружение, которое может быть как очень быстрым, так и очень медленным. Поэтому имитация поведения реактивной машины на самом деле ничего не добавляет, оставляя и то поведение, которое есть в генеративной машине. В общем понятно, что блокирующий переключатель не увеличивает мощность тестирования.

Обратно, поведение генеративной машины, когда хотя бы один переключатель находится в положении *free* и переключатели не меняют своего положения, имитируется в реактивной машине последовательностью нажатия одного и того же набора кнопок. Запрет всех внешних действий (все переключатели в положении *blocked*) Ван Глаббек предлагает имитировать в реактивной машине нажатием кнопки «лишнего» действия, про которое точно известно, что оно никогда не может выполняться. В алфавит `L` добавляется «лишнее» действие, а в машину – разрешающая его кнопка.

Нам достаточно показать, как поведение генеративной машины можно имитировать в параметризованной машине и наоборот. Поведение

генеративной машины имитируется в параметризованной машине быстрым нажатием той же самой кнопки после выполнения внешнего действия. Заметим, что, если оператор не успел достаточно быстро нажать кнопку, ничего страшного не случится, поскольку машина успеет выполнить только одно или несколько τ-действий, которые (*в машине без приоритетов*) она может выполнить и в том случае, когда кнопка была нажата немедленно. Иными словами, мы требуем, чтобы оператор мог работать быстро, но не заставляем его всегда работать быстро. Это согласуется с тем, что оператор должен моделировать любую скорость работы окружения.

Для имитации поведения параметризованной машины в генеративной машине можно использовать упоминаемую Ван Глаббеком модификацию генеративной машины, в которой переключатели автоматически сбрасываются в положении *blocked* каждый раз, когда выполняется внешнее действие. Это в точности соответствует в параметризованной машине тому, что после выполнения внешнего действия до нового нажатия какой-либо кнопки могут выполняться только τ-действия.

Для наблюдения внешних действий в генеративной и параметризованной машине имеется дисплей. Когда машина выполняет внешнее действие z, символ z высвечивается на дисплее. В параметризованной машине дисплей гаснет при нажатии следующей кнопки. Поскольку генеративная машина «непрерывного действия», в ней возможно наблюдение последовательности $\langle z, z, z, \ldots \rangle$ без изменения положения переключателей. Чтобы оператор мог различать разные выполнения одного действия, между двумя последовательными действиями дисплей должен гаснуть на короткий интервал времени t_0. В реактивной машине дисплея нет, но кнопка выполняемого действия «проваливается», машина приостанавливается и для продолжения тестирования нужно отжать «провалившуюся» кнопку и нажать ту же самую или новую кнопку (или несколько кнопок в модифицированной машине). Очевидно, эти различия несущественны.

Подводя итоги, отметим два важных момента (для определённости, в терминах параметризованной машины).

Переключение без наблюдения. В машине без приоритетов выполнимость τ-действий не зависит от множества разрешённых внешних действий (кнопок и переключателей). Поэтому нет смысла переключать кнопки без наблюдения: такое переключение не увеличивает мощность тестирования. Действительно, если была нажата кнопка “P”, а потом без наблюдения нажата другая кнопка “Q”, то в этом интервале времени машина могла выполнять только τ-действия. Но эти τ-действия она могла бы выполнять и в том случае, когда вместо кнопки “P” сразу нажималась кнопка “Q”, а второй раз, естественно, не нажималась. Тем самым, любое поведение (трассу внешних действий), которое можно наблюдать в первом случае, можно было бы наблюдать и во втором случае. В LTS-тесте запрет на переключение без наблюдения означает, что $t \xrightarrow{\tau}$ влечёт $\forall z \neq \tau \; t \xrightarrow{z} \not\Rightarrow$.

При наличии приоритетов переключение без наблюдения необходимо для полноты тестирования, поскольку различные множества разрешённых действий по-разному влияют на выполнение τ-действий, что приводит к внешне различимым поведениям. Например, если в реактивной системе приём стимула приоритетнее выдачи реакций и τ-действий, последние выполняются только тогда, когда тест не посылает ни одного стимула.

Тестовые истории и трассы наблюдений. Собирая полную информацию о процессе тестирования, оператор мог бы записывать последовательность выполняемых им действий и наблюдений. Такую последовательность нажимавшихся кнопок и наблюдаемых внешних действий будем называть *тестовой историей*, а её подпоследовательность наблюдений – *трассой наблюдений*, *наблюдаемой трассой* или просто *трассой*. В общем случае всё, что мы можем узнать при тестировании о реализации, сводится к множеству всех возможных тестовых историй. При наличии приоритетов нам недостаточно трасс, и приходится работать с историями.

Однако в машине без приоритетов выполнимость внешнего действия не зависит от того, какие ещё внешние действия разрешены. Если действие $z \in P$ наблюдалось после нажатия кнопки “P”, то оно точно также могло наблюдаться после нажатия любой другой кнопки “Q” такой, что $z \in Q$. Поэтому множество всех тестовых историй однозначно восстанавливается по множеству всех трасс наблюдений. Такое множество трасс будем называть *трассовой моделью реализации* или просто *трассовой реализацией*. Этой модели посвящена Глава 3. В LTS-модели $\mathbf{I}$ трасса определяется как последовательность внешних действий, которыми помечены переходы маршрута LTS, начинающегося в начальном состоянии, с пропуском τ-переходов. LTS-реализации различимы при взаимодействии только с точностью до их трассовой модели $\mathbf{I}=\boldsymbol{traces}(\mathbf{I})$. Поэтому спецификацию можно рассматривать как описание требований к множеству трасс реализации. Для выбранного нами типа конформности спецификация задаёт множество разрешаемых трасс, которому должны принадлежать все трассы конформной реализации. Сама спецификация может быть задана как множество трасс соответствующей LTS $\mathbf{\Sigma}=\boldsymbol{traces}(\mathbf{S})$; это множество трасс будем называть *трассовой моделью спецификации* или просто *трассовой спецификацией*. Предполагается, что трассовая спецификация однозначно определяет множество разрешаемых трасс, хотя, как увидим ниже, может с ним не совпадать.

Трассовая модель (как реализации, так и спецификации) является деревом (префикс-замкнута): если наблюдается (разрешается) некоторая трасса, то любой её префикс также наблюдается (разрешается). Это означает, что тестирование имеет смысл вести «до первой ошибки»: если после наблюдения разрешённой трассы σ наблюдается внешнее действие z, а трасса $\sigma \cdot \langle z \rangle$ запрещена, то тестирование можно завершить с вердиктом ***fail***, поскольку любое продолжение $\sigma \cdot \langle z \rangle \cdot \lambda$ также запрещено. Более того, как мы увидим ниже, продолжение тестирования в этой ситуации может быть «опасным».

2.5. Остановка машины и наблюдение отказов

Когда в машине нет выполнимых действий, она останавливается. Если оператор может это обнаруживать, то он получает новый вид наблюдения: машина стоит в состоянии, в котором не выполнимо ни одно действие, разрешённые действия образуют множество P. Такое множество P называется *отказом* (*refusal set*). Теперь в тестовые истории и трассы наблюдений мы будем помещать не только действия из алфавита L, но и наблюдаемые отказы как подмножества действий $P \subseteq L$. Впервые идея наблюдения отказов высказана в [137] и, независимо, в [111].

В машине без приоритетов остановка возможна только при отсутствии внутренней активности. LTS-реализация останавливается в состоянии s без τ-переходов $s \text{—} \tau \not\rightarrow$; такое состояние называется *стабильным*. Отказ P может наблюдаться в таком стабильном состоянии s, в котором $\forall z \in P \; s \text{—} z \not\rightarrow$. Правило «Переключение без наблюдения» соответственно корректируется: под наблюдениями понимаются не только внешние действия, но и отказы. Возможность восстановления тестовых историй по трассам с отказами также сохраняется, поскольку отказ P можно наблюдать только при нажатии кнопки “P”. Для получения трасс с отказами (*failure traces*) по LTS-реализации достаточно добавить отказы в алфавит внешних действий LTS, в каждом её стабильном состоянии добавить переходы-петли по наблюдаемым в этом состоянии отказам, и взять множество трасс внешних действий полученной LTS: $\mathbf{I} = \boldsymbol{Ftraces}(\mathbf{I})$.

Для наблюдения отказов в генеративной и реактивной машинах имеется *зелёная лампочка* (Рис.13), которая горит, если машина выполняет какое-либо внешнее или внутреннее действие, и гаснет, когда машина останавливается. Отказ вычисляется как множество действий, написанных на переключателях в положении *free* в генеративной машине, или на тех кнопках, которые нажимает оператор в реактивной машине. Возможен другой режим работы зелёной

лампочки, когда она горит только при выполнении внутренних действий. Если зелёная лампочка гаснет, то это либо остановка, либо выполнение внешнего действия. В реактивной машине остановка обнаруживается, если кнопка не проваливается. В генеративной машине – если дисплей погашен в течение времени $t>t_0$, чтобы убедиться, что это остановка, а не промежуток между выполнением двух внешних действий.

Зелёная лампочка

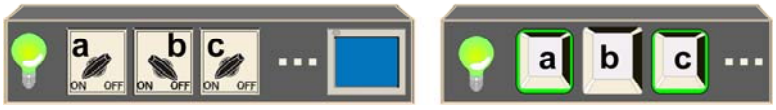


Рис.13.

Хотя для обнаружения остановки годятся оба режима работы зелёной лампочки, они различаются мощностью тестирования. Дело в том, что второй режим позволяет делать отдельное наблюдение внутренней активности (непустой последовательности внутренних действий), когда зелёная лампочка горит, а дисплей пуст. Заметим, что сами внутренние действия, по-прежнему, неразличимы между собой, в том числе неразличимы одно τ-действие и любая конечная последовательность τ-действий. Если наблюдение внутренней активности обозначить τ, то различаются трассы $\langle z,\tau,z`\rangle$ и $\langle z,z`\rangle$, а также трассы $\langle z,\tau,P\rangle$ и $\langle z,P\rangle$, где $z,z`\in L$ и $P\subseteq L$, которые в первом режиме не различимы. Что касается практичности такой тестовой возможности, то это зависит от условий взаимодействия. Например, когда мы запускаем программу на своём компьютере, а она долго не отвечает, мы можем «подсмотреть», тратит ли программа процессорное время, изменяет ли какие-то переменные или файлы. С другой стороны, если программа предназначена для удалённого взаимодействия, то при соответствующем удалённом тестировании такой возможности нет. Однако более важный вопрос заключается в том, нужна ли нам такая тестовая возможность? Трудно представить себе ситуации, когда

наличие или отсутствие конечной внутренней активности между внешним действием (или началом работы) и следующим внешним действием или отказом должно трактоваться как ошибка.

Для обнаружения остановки вовсе не обязательна возможность наблюдать (внутреннюю) активность реализации. Например, пусть время выполнения любого внешнего действия и любой конечной внутренней активности ограничено сверху известным значением t_1. Тогда истечение тайм-аута $t>t_1$ при отсутствии внешних действий означает либо дивергенцию, либо остановку машины. Если мы уверены, что дивергенции нет, то остаётся только остановка. В реактивных системах такой тайм-аут может устанавливаться при приёме всех реакций без посылки стимулов. Соответствующий отказ называют *стационарностью* – это множество всех реакций, обозначаемое символом δ [149,150]. В LTS-реализации такой отказ наблюдается в *стационарном состоянии* (*quiescent state*) – стабильном состоянии, в котором нет переходов по реакциям. Если посылка одного стимула `?x` предусматривает достоверное (безошибочное и, значит, не требующее тестирования) «уведомление о вручении», которое должно придти за время, ограниченное сверху известным значением, то при отсутствии дивергенции мы можем наблюдать отказ `{?x}`, называемый *блокировкой стимула* (*input refusal*). В LTS-реализации блокировка `{?x}` наблюдается в стабильном состоянии, в котором нет перехода по стимулу `?x`. Подчеркнём, что истечение тайм-аута само по себе не позволяет различить дивергенцию и остановку машины. Наблюдение остановки возможно только при условии, что в машине нет дивергенции. Эту проблему мы рассмотрим позже.

Другой вариант: реализация может сама изнутри чёрного ящика сообщать машине не только о выполнении внешнего действия, но и об остановке. Например, для наблюдения стационарности вместо тайм-аута можно использовать достоверное «уведомление о прекращении передачи реакций». Для наблюдения блокировки стимула вместо тайм-аута можно использовать не

только достоверное «уведомление о вручении», но и достоверное «уведомление о невручении».

В параметризованной машине мы предлагаем абстрагироваться от способа обнаружения остановки. Будем считать, что, если при нажатии кнопки “P” может быть обнаружена остановка, то это делает сама машина и высвечивает на дисплей специальный символ θ (Рис.14). (Здесь мы следуем [111].)

Рис.14.

В LTS-тесте этому соответствует добавление в алфавит символа θ и возможность определять θ-переходы. При композиции θ-переход в тесте выполним тогда и только тогда, когда никакие переходы в реализации не выполнимы. Для машины без приоритетов это означает, что в реализации нет τ-перехода и нет перехода по внешнему действию z, если в тесте есть переход по противоположному действию $\underline{z}$. Это можно записать дополнительным правилом вывода: ***Stop*** $\& \; t \text{—}\theta\rightarrow t` \vdash st \text{—}\tau\rightarrow st`$, где

Stop $=_{def} s \text{—}\tau\not\rightarrow \; \& \; \forall z \in L \cap \underline{M} \; (s \text{—}z\not\rightarrow \vee t \text{—}\underline{z}\not\rightarrow)$.

Поскольку мы рассматриваем тестирование, в котором нет переключения кнопок без наблюдения, в LTS-тесте $t \text{—}\tau\rightarrow$ влечёт не только $t \text{—}z\not\rightarrow$ для каждого внешнего действия z, но и $t \text{—}\theta\not\rightarrow$. В таком LTS-тесте τ-переход может быть только в таком состоянии, в котором определены только τ-переходы, что соответствует отсутствию нажатой кнопки. Поэтому θ-переход выполним тогда и только тогда, когда никакие другие переходы не выполнимы. Это совпадает с традиционной семантикой θ-перехода, которая записывается следующим дополнительным правилом:

Deadlock $\& \; t \text{—}\theta\!\rightarrow t` \vdash \; st \text{—}\tau\!\rightarrow st`$, где

Deadlock $=_{def} \; s \text{—}\tau\!\nrightarrow \; \& \; t \text{—}\tau\!\nrightarrow \; \& \; \forall z \in L \cap \underline{M} \; (s \text{—} z\!\nrightarrow \vee \; t \text{—} \underline{z}\!\nrightarrow)$.

Для замкнутой системы <реализация-тест> имеем $M=\underline{L}$ и $L\lceil\rfloor M=\varnothing$. Поэтому при остановке реализации в состоянии s возможен любой отказ $P \subseteq \{z \in L \mid s \text{—} z\!\nrightarrow\}$. В машине, параметризованной семейством $\mathfrak{R}$, должно быть также $P \in \mathfrak{R}$.

Теперь уточним параметризацию нашей машины, учитывая отказы. Мы будем считать, что одни отказы могут наблюдаться при остановке, а другие нет. Например, в реактивных системах у нас может быть тайм-аут ожидания реакций, но не быть никакого уведомления о вручении или невручении стимула. В этом случае стационарность наблюдается (при отсутствии дивергенции), а блокировки стимулов не наблюдаются. Такие реактивные системы рассматриваются, например, для отношений ***iot***, ***ioconf***, ***ior*** и ***ioco***, которые мы рассмотрим ниже.

Таким образом, семейство кнопок разбивается на два подсемейства $\mathfrak{R}$ – кнопки с наблюдаемыми отказами, и $\mathfrak{Q}$ – кнопки с ненаблюдаемыми отказами, суммарно покрывающие алфавит: $\cup\mathfrak{R}\cup\cup\mathfrak{Q}=L$. Будем считать, что $\mathfrak{R}\cap\mathfrak{Q}=\varnothing$, поскольку при наличии кнопки “P” с наблюдаемым отказом добавление кнопки “P” с ненаблюдаемым отказом не увеличивает мощность тестирования. Машину, параметризованную семействами $\mathfrak{R}$ и $\mathfrak{Q}$ будем называть $\mathfrak{R}/\mathfrak{Q}$-машиной и говорить, что она определяет тестовую $\mathfrak{R}/\mathfrak{Q}$-семантику взаимодействия. Трассы с отказами из $\mathfrak{R}$ будем называть $\mathfrak{R}$-трассами, а множество таких трасс LTS – $\mathfrak{R}$-моделью.

Для $\mathfrak{R}=\mathcal{P}(\mathtt{L})$ мы будем говорить о ***F-трассах*** и ***F-модели***. Для реализации её ***F***-модель – это множество трасс наблюдений на $\mathcal{P}(\mathtt{L})/\varnothing$-машине, которую будем называть ***F-машиной***. Для $\mathfrak{R}\subseteq\mathcal{P}(\mathtt{L})$ и ***F***-модели $\mathbf{T}$ подмножество её $\mathfrak{R}$-трасс является $\mathfrak{R}$-моделью, которую будем обозначать $\mathbf{T}^{\downarrow}\mathtt{L}_{\mathfrak{R}}=\mathbf{T}\cap(\mathtt{L}\cup\mathfrak{R})^{*}$. Верно и обратное: любая $\mathfrak{R}$-модель является подмножеством $\mathfrak{R}$-трасс некоторой ***F***-модели. Далее реализацию и спецификацию будем понимать как ***F***-модели и обозначать как $\mathbf{I}$ и $\mathbf{\Sigma}$, соответственно. По LTS-моделям строятся соответствующие ***F***-модели взятием всех их трасс со всеми отказами $\mathbf{I}=\boldsymbol{Ftraces}(\mathtt{I})$ и $\mathbf{\Sigma}=\boldsymbol{Ftraces}(\mathtt{S})$. Как по спецификации $\mathbf{\Sigma}$ определяется множество разрешённых трасс, мы увидим ниже.

Отметим, что Ван Глаббек и Милнер не рассматривали таких вариантов своих машин: у них зелёная лампочка либо есть и все отказы наблюдаемы, либо нет и все отказы не наблюдаемы. Соответственно, множество наблюдаемых трасс: либо ***F***-модель=$\mathcal{P}(\mathtt{L})$-модель (все подмножества алфавита являются наблюдаемыми отказами), либо $\varnothing$-модель (нет наблюдаемых отказов, трассы содержат только внешние действия).

2.6. Взаимодействие теста и реализации. Безопасное тестирование

Остановка машины может вызывать или не вызывать тупик при взаимодействии. Для определённости будем рассматривать $\mathfrak{R}/\mathfrak{Q}$-машину. Тупик не возникает в двух случаях. 1) Окружение может продолжить работу не дожидаясь, чем закончится последний шаг взаимодействия: внешним действием или отказом, и закончится ли вообще. Оператор переключает кнопку при отсутствии наблюдений. В состоянии LTS-окружения/теста определён τ-

переход. 2) При остановке возникает наблюдаемый отказ и окружение реагирует на него. Оператор наблюдает отказ, после чего нажимает (другую) кнопку. Состояние LTS-окружения/теста стабильно и в нём определён θ-переход.

Возникновение тупика можно считать нормальным завершением взаимодействия, если это совпадает с завершением работы окружения. Оператор не нажимает и не будет нажимать кнопки: все внешние действия запрещены, никаких наблюдений больше не будет. LTS-окружение (тест) находится в терминальном состоянии (состоянии без переходов). При тестировании выносится вердикт ***pass*** или ***fail***.

Замечание о переключении кнопок при отсутствии наблюдений.

Мы говорили о том, что переключение с кнопки “Q” на кнопку “P” при отсутствии наблюдений не увеличивает мощность тестирования. Это верно лишь в том смысле, что, если после нажатия кнопки “Q” наблюдения не будет, то можно было бы сразу нажимать кнопку “P”. Однако, если после нажатия кнопки “Q” возможно наблюдение действия $q \in Q$, то для получения всех трасс мы должны нажимать какую-нибудь кнопку, разрешающую действие q. В частности, если это действие разрешается только кнопкой “Q”, то мы должны обязательно нажимать эту кнопку. Если это Ω-кнопка и в реализации возможен ненаблюдаемый отказ Q, то возможно отсутствие наблюдений. Проблема в том, что мы не знаем, будет ли наблюдаться действие $q \in Q$ или никаких наблюдений не будет.

Можно было бы бесконечное число раз прогонять тест с возрастающей последовательностью интервалов времени ожидания наблюдений, переключая кнопки при истечении текущего интервала. Тогда, если в реализации есть действие $q \in Q$, то на каком-то прогоне теста мы будем его наблюдать. Отказ Q становится отрицательным наблюдением: он означает,

что во всех прогонах теста нам пришлось сделать переключение кнопок, поскольку не наблюдались действия из множества Q. Отрицательные наблюдения мы рассмотрим ниже.

Здесь нам важно то, что на практике приходится всегда иметь дело только с конечным числом прогонов теста. Поэтому мы должны принять реализационную гипотезу, позволяющую обходиться конечным числом прогонов. Такая гипотеза, фактически, сводится к ограничению времени ожидания наблюдения: если в течение соответствующего тайм-аута наблюдений нет, то их и не будет. Но тогда реализуется наблюдение отказа Q как истечение тайм-аута при нажатии кнопки “Q”.

Альтернативные подходы связаны с явным введением в семантику взаимодействия времени или вероятностей [57,63,98]. Пример введения времени: в спецификации указывается, что действие $q \in Q$ либо должно наблюдаться до истечения контрольного времени t после нажатия кнопки “Q”, либо не важно, есть оно в реализации или нет. Здесь имеется в виду, что, если действие в реализации наблюдается после истечения контрольного времени, то спецификация разрешает любое поведение после этого действия. Пример введения вероятности: в спецификации указывается, что действие $q \in Q$ может наблюдаться с некоторой вероятностью. Предполагается, что распределение вероятности по прогонам теста известно (например, равномерно). Тогда можно прогонять тест конечное число раз и определять конформность с некоторой вероятностью.

Во всех такого рода альтернативных подходах, кроме наблюдения действий или их отсутствия, вводятся дополнительные понятия времени или вероятности. Это вызывает необходимость более сложного понимания взаимодействия и использования более сложных моделей реализации и спецификации. В данной работе такие альтернативные подходы не рассматриваются. Мы будем считать, что переключение кнопок при отсутствии наблюдений не выполняется.

Тупик в нетерминальном состоянии окружения мы будем рассматривать как ошибку взаимодействия: окружение/тест «желает» продолжения, но оно невозможно. Заметим, что в этом случае нажата какая-то кнопка. «Виновником» такой ошибки может быть как реализация, так и окружение. Допустим, что реализация «не виновата». Тогда, если отказ наблюдаемый, то окружение «виновато» в том, что не реагирует на него, хотя могло бы это делать. Если отказ ненаблюдаемый, то «вина» окружения заключается в том, что оно позволило этому произойти: оператор не должен был нажимать кнопку "P", если отказ P ненаблюдаемый и может возникнуть в данной ситуации. Если окружение всё делает «правильно», а ошибка взаимодействия возникает, то «виновата» реализация: нажимаемая кнопка не позволяет наблюдать отказ, а реализация останавливается. Проблема, однако, в том, что ошибка взаимодействия, по определению, не проверяема при тестировании: возможность проверки как раз означала бы, что отказ наблюдаем.

Если нажатие кнопки "P" после трассы σ не может вызывать ошибки взаимодействия (ненаблюдаемого отказа), то будем называть такую кнопку *безопасной в реализации после трассы* σ. Это означает, что в ***F***-модели реализации $\mathbf{I}$ трасса $\sigma \in \mathbf{I}^{\downarrow L_{\mathfrak{R}}}$ не продолжается отказом P, если $P \in \mathfrak{Q}$ (в LTS-реализации трасса σ не заканчивается в стабильном состоянии, в котором нет переходов по действиям из P):

P ***safe in*** $\mathbf{I}$ ***after*** σ $=_{\text{def}}$ $P \in \mathfrak{R} \vee \sigma \cdot \langle P \rangle \notin \mathbf{I}$.

Протоколом взаимодействия будем называть правило, определяющее, какие $\mathfrak{Q}$-кнопки после каких трасс наблюдений разрешено нажимать, а какие нет. Первые будем называть *безопасными*, а вторые – *опасными после трассы в данном протоколе*. $\mathfrak{R}$-кнопки объявляются безопасными (в смысле отсутствия ненаблюдаемых отказов) после любой трассы. Таких протоколов может быть много. Нас интересует не любое взаимодействие, а только то, которое происходит по правилам некоторого заданного i-го протокола. Будем говорить,

что реализация отвечает i-ой *гипотезе о безопасности*, если при взаимодействии с ней любого окружения, выполняющего правила i-го протокола, не возникает ошибок взаимодействия. Каждая i-ая гипотеза о безопасности определяет свой i-ый класс *безопасных* (безопасно тестируемых) реализаций. Безопасность реализации в заданном i-ом протоколе невозможно проверить при тестировании, поскольку такая проверка означала бы проверку наличия или отсутствия ошибки взаимодействия. Поэтому соответствующая гипотеза о безопасности является предусловием тестирования. В свою очередь, тест (оператор машины) должен соблюдать i-ый протокол взаимодействия, и такое тестирование будем называть *безопасным*.

Только в этом контексте можно говорить о конформности или неконформности реализации: каждый i-ый протокол взаимодействия определяет свой, i-ый класс конформных реализаций как подкласс i-ого класса безопасных реализаций. Цель безопасного тестирования по i-ому протоколу – проверка конформности реализации при условии, что реализация взята из i-го класса безопасных реализаций. Заметим, что реализация, конформная в одном протоколе, может оказаться не безопасной в другом протоколе.

Теперь мы должны уточнить определение конформности как вложенности наблюдаемых трасс реализации во множество трасс, разрешаемых спецификацией. Фактически, конформность параметризуется заданным i-ым протоколом взаимодействия. Во-первых, нас интересуют не все реализации, а только те, которые удовлетворяют i-ой гипотезе о безопасности. Во-вторых, нас интересуют не все трассы таких реализаций, которые могут наблюдаться на данной $\mathfrak{R}/\mathfrak{Q}$-машине, а только те, которые наблюдаемы при взаимодействии по i-ому протоколу. Во множестве таких трасс спецификация выделяет подмножество разрешённых трасс. Теперь разрешённые трассы – это не все трассы, которые могут быть в конформных реализациях, а только те из них, которые могут наблюдаться при безопасном тестировании. Мы будем считать,

что спецификации соответствует ***F***-модель $\mathbf{\Sigma}$, и множество разрешённых трасс одно и то же в любом протоколе взаимодействия – это множество $\mathfrak{R}$-трасс $\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}}$, оно является подмножеством трасс, которые могут наблюдаться при безопасном тестировании. Остальные трассы, которые могут быть в безопасных и конформных реализациях, но не наблюдаются при безопасном тестировании, определяются протоколом. Можно также считать, что спецификация – это пара (***F***-модель, протокол взаимодействия). Для того, чтобы множество разрешённых трасс совпадало с $\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}}$, мы ограничиваемся только такими протоколами, которые удовлетворяют следующим правилам.

Первое правило: Все $\mathfrak{R}$-кнопки объявляются безопасными после любой $\mathfrak{R}$-трассы.

Второе правило: Все трассы из $\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}}$ могут быть проверены при тестировании. Это означает: если трасса $\sigma\cdot\langle z\rangle\in\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}}$, то протокол должен объявить безопасной после σ *хотя бы одну* кнопку “P”, разрешающую действие z, то есть $z\in P$. Заметим, что $\mathfrak{R}$-кнопки безопасны по любому протоколу, но мы не делаем ограничений сверху на множество безопасных $\mathfrak{Q}$-кнопок. В частности, можно все $\mathfrak{Q}$-кнопки, разрешающие действие z, объявить безопасными после трассы σ. Это правило позволяет проверить, продолжается ли в реализации трасса σ внешним действием z или нет. Если после нажатия кнопки “P” наблюдается действие z, то тестирование можно продолжить, чтобы сравнить поведение реализации и спецификации после трассы $\sigma\cdot\langle z\rangle$. Если наблюдается действие $z`$, которого нет в спецификации $\sigma\cdot\langle z`\rangle\notin\mathbf{\Sigma}$, то это ошибка (неконформность).

Третье правило: Протокол не налагает на реализацию $\mathbf{I}$ лишних ограничений по безопасности, то есть все $\mathfrak{Q}$-кнопки, которые протокол не объявил безопасными по первому и второму правилам, объявляются опасными и, тем самым, гипотеза о безопасности разрешает им быть опасными в реализации. В частности, если в спецификации трасса σ не продолжается действиями, разрешаемыми некоторой $\mathfrak{Q}$-кнопкой "$\mathtt{Q}$", то есть $\forall \mathtt{z}\in\mathtt{Q}\ \ \sigma\cdot\langle \mathtt{z}\rangle\notin\mathbf{\Sigma}$, то кнопка "$\mathtt{Q}$" опасна после трассы σ по любому протоколу. Заметим, что, если бы протокол объявил безопасной такую кнопку, то любая безопасная реализация, имеющая разрешённую трассу σ, оказалась бы неконформной.

Будем задавать протокол взаимодействия в форме отношения «кнопка безопасна после $\mathfrak{R}$-трассы спецификационной модели», которое должно отвечать следующим правилам:

$\forall\sigma\in\mathbf{\Sigma}^{\downarrow}\mathtt{L}_{\mathfrak{R}}\ \ \forall\mathtt{R}\in\mathfrak{R}\ \ \forall\mathtt{z}\in\mathtt{L}\ \ \forall\mathtt{Q}\in\mathfrak{Q}$

1) $\mathtt{R}$ ***safe by*** $\mathbf{\Sigma}$ ***after*** σ,

2) $\sigma\cdot\langle \mathtt{z}\rangle\in\mathbf{\Sigma}\ \Rightarrow\ \exists\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}\ \ \mathtt{z}\in\mathtt{P}\ \ \&\ \ \mathtt{P}$ ***safe by*** $\mathbf{\Sigma}$ ***after*** σ,

3) $\mathtt{Q}$ ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma\ \Rightarrow\ \exists\mathtt{v}\in\mathtt{Q}\ \ \sigma\cdot\langle \mathtt{v}\rangle\in\mathbf{\Sigma}$.

Теперь, кроме разрешённой трассы $\sigma\in\mathbf{\Sigma}^{\downarrow}\mathtt{L}_{\mathfrak{R}}\cap\mathbf{I}$, в конформной реализации могут быть ненаблюдаемые по данному протоколу трассы. Это непосредственно следует из того, что опасные кнопки не нажимаются при безопасном тестировании и, следовательно, наличие или отсутствие некоторых продолжений трассы σ не может быть проверено.

Во-первых, трасса σ может продолжаться в реализации ненаблюдаемым отказом $\mathtt{Q}\in\mathfrak{Q}$, если кнопка "$\mathtt{Q}$" объявлена протоколом опасной после σ и не нарушаются гипотеза о безопасности и условие конформности. Это значит, что

любая кнопка “P”, объявленная безопасной после σ, либо разрешает хотя бы одно действие, которое не разрешает кнопка “Q” и которым продолжается трасса σ в спецификации и реализации, либо это $\mathfrak{R}$-кнопка и отказ P продолжает трассу σ в спецификации и реализации: $\forall P \in \mathfrak{R} \cup \mathfrak{Q}\ \exists u\ (u \in P \setminus Q \vee u = P \in \mathfrak{R})\ \&\ \sigma \cdot \langle u \rangle \in \mathbf{I} \cap \mathbf{\Sigma}$.

Во-вторых, допускается любая трасса вида $\sigma \cdot \langle z \rangle \cdot \lambda$, где внешнее действие z разрешается только кнопками, объявленными опасными после σ.

В качестве примера рассмотрим $\mathfrak{R}/\mathfrak{Q}$-машину с $\mathfrak{R}=\varnothing$ и $\mathfrak{Q}=\{\{a,b\},\{b,c\}\}$ и спецификационную модель $\mathbf{\Sigma}^{\downarrow L_{\mathfrak{R}}}=\{\epsilon, \langle b \rangle\}$. Возможны три типа протокола взаимодействия в зависимости от того, какие кнопки считаются безопасными в самом начале (после пустой трассы): 1) только “{a,b}”, 2) только “{b,c}”, 3) любые. Реализация $\mathbf{I}$ безопасна, если: 1) $\langle b \rangle \in \mathbf{I} \vee \langle a \rangle \in \mathbf{I}$, 2) $\langle b \rangle \in \mathbf{I} \vee \langle c \rangle \in \mathbf{I}$, 3) $\langle b \rangle \in \mathbf{I} \vee \langle a \rangle \in \mathbf{I}\ \&\ \langle c \rangle \in \mathbf{I}$. Реализация конформна, если: 1) $\langle b \rangle \in \mathbf{I}\ \&\ \langle a \rangle \notin \mathbf{I}$, 2) $\langle b \rangle \in \mathbf{I}\ \&\ \langle c \rangle \notin \mathbf{I}$, 3) $\langle b \rangle \in \mathbf{I}\ \&\ \langle a \rangle \notin \mathbf{I}\ \&\ \langle c \rangle \notin \mathbf{I}$. В этом примере в любой конформной реализации вместе с любой *непустой* трассой σ может быть любое продолжение $\sigma \cdot \lambda$.

2.7. Гипотеза о безопасности и безопасная конформность

Безопасной трассой реализации будем называть её $\mathfrak{R}$-трассу, в которой каждое встречающееся внешнее действие z разрешается хотя бы одной кнопкой, безопасной в реализации после префикса трассы, непосредственно предшествующего этому действию:

$\mathit{SafeIn}(\mathbf{I}) =_{\text{def}} \{\sigma \in \mathbf{I}^{\downarrow}\mathbb{L}_{\mathfrak{R}} \mid \forall \mu \ \forall z \in \mathbb{L}$

$(\mu \cdot \langle z \rangle \le \sigma \Rightarrow \exists P \in \mathfrak{R} \cup \mathfrak{Q} \ P \ \mathit{safe\ in}\ \mathbf{I}\ \mathit{after}\ \mu \ \&\ z \in P\})$.

Гипотеза о безопасности требует: если $\mathfrak{R}$-трасса σ спецификации безопасна в реализации, то любая кнопка, безопасная после σ в спецификации, безопасна после σ в реализации:

$\mathbf{I}\ \mathit{safe\ for}\ \mathbf{\Sigma} =_{\text{def}} \forall \sigma \in \mathbf{\Sigma}^{\downarrow}\mathbb{L}_{\mathfrak{R}} \cap \mathit{SafeIn}(\mathbf{I}) \ \forall P \in \mathfrak{R} \cup \mathfrak{Q}$

$(P\ \mathit{safe\ by}\ \mathbf{\Sigma}\ \mathit{after}\ \sigma \Rightarrow P\ \mathit{safe\ in}\ \mathbf{I}\ \mathit{after}\ \sigma)$.

Тестирование ведётся «до первой ошибки»: прекращается сразу после обнаружения неконформности. Теперь продолжение тестирования после получения запрещённой трассы не только бессмысленно (не влияет на вердикт), но и небезопасно, поскольку гипотеза о безопасности уже не даёт никаких гарантий.

Обозначим множество внешних действий и отказов, наблюдаемых в модели T после трассы σ при нажатии кнопки P:

$\mathit{obs}(\sigma, P, T) =_{\text{def}} \{z \mid \sigma \cdot \langle z \rangle \in T \ \&\ (z \in P \vee z = P \ \&\ P \in \mathfrak{R})\}$.

Будем говорить, что реализация *безопасно конформна* спецификации, если она безопасна и любое наблюдение в ответ на нажатие безопасной кнопки, разрешается спецификацией. Такую конформность будем обозначать ***saco*** (***SA****fe* ***CO****nformance*).

$\mathbf{I}\ \mathit{saco}\ \mathbf{\Sigma} =_{\text{def}} \mathbf{I}\ \mathit{safe\ for}\ \mathbf{\Sigma}\ \&$

$\forall \sigma \in \mathbf{\Sigma} \cap \mathit{SafeIn}(\mathbf{I}) \ \forall P\ \mathit{safe\ by}\ \mathbf{\Sigma}\ \mathit{after}\ \sigma\ \mathit{obs}(\sigma, P, \mathbf{I}) \subseteq \mathit{obs}(\sigma, P, \mathbf{\Sigma})$.

Генерация полного набора тестов для отношения ***saco*** сводится к тому, чтобы после каждой трассы σ, имеющейся в спецификации и наблюдаемой в реализации, нажать каждую кнопку “P”, безопасную после этой трассы. Если после этого наблюдается действие $z \in P$ или отказ $P \in \mathfrak{R}$, продолжающие трассу σ в спецификации, то тестирование либо заканчивается с вердиктом

pass, либо продолжается нажатием очередной кнопки. В противном случае тест выносит вердикт ***fail***.

Все определения безопасности и конформности, сделанные выше, применяются к LTS-моделям с помощью взятия их ***F***-моделей. Например, $\mathbf{I}$ ***saco*** $\mathbf{S}$ $=_{\text{def}}$***Ftraces***$(\mathbf{I})$ ***saco*** ***Ftraces***$(\mathbf{S})$. Формально трассовая и LTS-модели рассматриваются в главах 3 и 4, соответственно.

2.8. Пополнение спецификаций

Теперь обратим внимание на различие в определении безопасных кнопок для реализации и спецификации: ***safe in*** и ***safe by***. В реализации $\mathfrak{Q}$-кнопка "$\mathtt{P}$" безопасна после трассы, если трасса не продолжается отказом $\mathtt{P}$, а в спецификации – если трасса продолжается хотя бы одним действием $\mathtt{z}\in\mathtt{P}$ (да и то не обязательно, если есть другие безопасные кнопки, разрешающие все действия, которые продолжают трассу в спецификации и разрешаются кнопкой "$\mathtt{P}$"). В первом случае трасса не может продолжаться отказом $\mathtt{P}$, а во втором случае – должна продолжаться действием $\mathtt{z}\in\mathtt{P}$, но может продолжаться и отказом $\mathtt{P}$. Это означает, что мы игнорируем $\mathfrak{Q}$-отказы в спецификации после трассы, которая продолжается действиями из этого отказа. Фактически, во всех определениях мы используем для реализации $\mathfrak{R}\cup\mathfrak{Q}$-проекцию $\mathbf{I}^{\downarrow}\mathrm{L}_{\mathfrak{R}\cup\mathfrak{Q}}$, а для спецификации $\mathfrak{R}$-проекцию $\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}}$.

Это различие является причиной серьёзных проблем: нерефлексивности и немонотонности конформности.

Сначала рассмотрим первую проблему: одна и та же модель, рассматриваемая как реализация, может быть не безопасна и, следовательно, неконформна самой себе, рассматриваемой как спецификация. Более того, в спецификации могут быть $\mathfrak{R}$-трассы, которых не может быть ни в какой конформной реализации. Пример приведён на Рис.15 для реактивных LTS с

семантикой отношения ***ioco***. Здесь в LTS-спецификации **s** кнопка "?x" безопасна после трассы ⟨?x⟩ и, следовательно, после трассы ⟨δ,?x,δ⟩ (отказ не меняет состояния LTS). Поэтому, если в реализации есть трасса ⟨δ,?x,δ⟩, то в ней есть и трасса ⟨δ,?x,δ,?x⟩. Поскольку приём реакций безопасен, должна наблюдаться хотя бы одна из трасс ⟨δ,?x,δ,?x,!a⟩, ⟨δ,?x,δ,?x,!b⟩ или ⟨δ,?x,δ,?x,δ⟩. Но тогда в реализации есть хотя бы одна из трасс ⟨?x,δ,?x,!a⟩, ⟨δ,?x,?x,!b⟩ или ⟨?x,?x,δ⟩. Каждая из этих трасс является продолжением безопасной трассы спецификации наблюдением, порождаемым безопасной кнопкой "δ", но отсутствующим в спецификации, что противоречит конформности.

Неконформная трасса спецификации

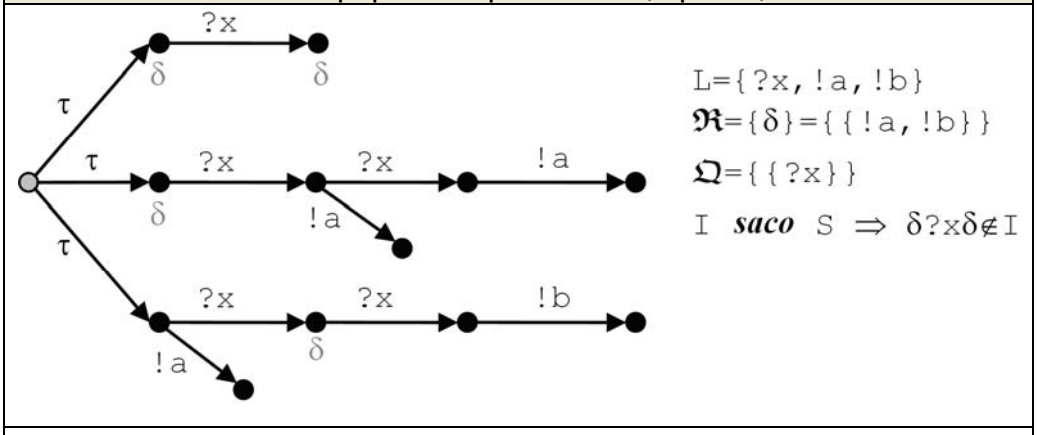


Рис.15.

Эта *нерефлексивность* прямо противоречит интуиции: если реализацию «списать» со спецификации, то мы получим заведомо ошибочную реализацию. Заметим, что, если $\mathfrak{Q}=\varnothing$, то ***safe by*** = ***safe in***, и отношение ***saco*** рефлексивно (и транзитивно, то есть является предпорядком). Это наводит на мысль перейти от $\mathfrak{R}/\mathfrak{Q}$-семантики к $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантике, когда все отказы наблюдаемы. Переход делается с помощью преобразования спецификации C, которое называется *пополнением*. Его можно определить как для трассовой, так

и для LTS-модели. В обоих случаях должна сохраняться конформность, то есть множество конформных реализаций. Для трассовой модели: $\forall \mathbf{I}\ \mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}\ \boldsymbol{\Sigma} \Leftrightarrow \mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ C(\boldsymbol{\Sigma})$. Здесь в нижнем индексе мы указали, для какой тестовой семантики рассматривается конформность. LTS-пополнение определяется аналогично, но, поскольку одной трассовой модели может соответствовать несколько LTS с тем же множеством трасс, оно не единственно.

В пополненной спецификации $C(\boldsymbol{\Sigma})$ могут появиться новые трассы, если они допустимы в конформных реализациях. Такого рода пополнение аналогично, так называемому, *демоническому пополнению* для реактивных систем без наблюдаемых блокировок стимулов [66,67,105]. Отличие в том, что мы в пополненной части модели допускаем любые отказы, в том числе (ненаблюдаемые) блокировки. Кроме того, в [66,67] предлагается LTS-преобразование, которое не полностью сохраняет конформность ***ioco*** (ослабляет её). В [105] предлагается трассовое преобразование, сохраняющее конформность, но этой конформностью является не ***ioco***, а ***ioconf***. Эти и некоторые другие конформности, встречающиеся в литературе, мы рассмотрим ниже. Обзор различных видов пополнений можно найти в [71,113].

Заметим, что пополненную спецификацию уже нельзя рассматривать в $\mathfrak{R}/\mathfrak{Q}$-семантике, поскольку в ней она не эквивалентна исходной спецификации (пример на Рис.16).

Пополненная спецификация
не эквивалентна исходной спецификации в $\mathfrak{R}/\mathfrak{Q}$-семантике

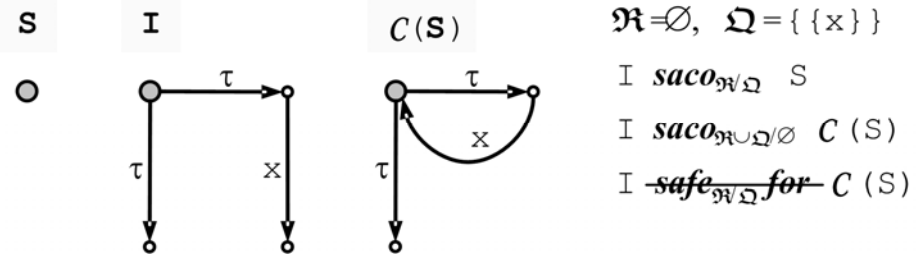


Рис.16.

Для доказательства существования трассового пополнения достаточно взять объединение всех конформных трассовых реализаций: $\mathcal{C}(\Sigma) = \cup\{\mathbf{I} \mid \mathbf{I}\ \textit{saco}\ \Sigma\}$. Для LTS-пополнения используется LTS, множество трасс которой совпадает с объединением всех конформных трассовых реализаций. Конечно, это преобразование не алгоритмизуемо, и можно поставить задачу нахождения алгоритмизуемого пополнения. Решению проблемы пополнения спецификации посвящена Глава 5, в которой данная тема рассматривается формально.

2.9. Монотонность конформности

Вторая проблема – это, так называемая, *немонотонность конформности*, то есть несохранение конформности при композиции LTS: композиция реализаций, конформных своим спецификациям, оказывается неконформной композиции этих спецификаций. Эта проблема не может быть поставлена в теории $\mathfrak{R}$-трасс, поскольку там не определена композиция трассовых моделей.

Проблема монотонности носит более общий характер, чем проблема пополнения, и остаётся даже в том случае, когда нет $\mathfrak{Q}$-кнопок.

Частным случаем проблемы композиции является асинхронное тестирование или тестирование в контексте. Такое тестирование можно рассматривать как тестирование системы из двух компонентов, один из которых – реализация, а другой – фиксированная среда взаимодействия. Этот частный случай очень важен, поскольку практическое тестирование почти всегда является асинхронным, особенно, если тестируется система в процессе её реальной работы.

Решением проблемы является *монотонное* LTS-преобразование $\mathcal{M}$, которое обладает двумя свойствами: 1) сохраняет конформность реализаций, и 2) композиция реализаций, конформных своим *преобразованным* спецификациям, конформна композиции этих *преобразованных* спецификаций. Заметим, что при асинхронном тестировании преобразованию подлежит только

реализация, так как среда взаимодействия фиксирована. Естественно, композиция LTS-пополнения и монотонного LTS-преобразования $\mathcal{MC}$ является LTS-пополнением, которое можно назвать *монотонным пополнением*. Проблеме монотонности и, в частности, алгоритмизуемому монотонному преобразованию спецификации посвящена Глава 6, в которой данная тема рассматривается формально.

2.10. Разрушение

Мы вводим новый вид действия машины, которое называем *разрушением* и обозначаем символом γ. Под γ-действием понимается любое нежелательное (недекларированное или запрещённое) поведение системы, в том числе и реальное разрушение системы, которого нельзя допускать при тестировании (например, нарушение предусловия вызова программы или кнопка немедленного саморазрушения для систем оборонного назначения). Разрушение, как и отказ, является одним из способов интерпретации неспецифицированного поведения (например, для автоматов Мили [71,113]). Отличие в том, что отказ предполагает «защиту» системы от нежелательных обращений, а разрушение ничего не гарантирует.

Исключение из рассмотрения обращений, разрушающих реализацию, часто мотивируют тем, что реализация должна проверять корректность параметров обращения [96,149]. Если параметры некорректны, реализация либо игнорирует обращение, либо сообщает об ошибке. 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 обращений с параметрами сложной внутренней структуры 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 вызова операций [99]. Например, вызов операции освобождения участка памяти, который не был ранее получен через операцию запроса памяти, является нарушением предусловия. Тестированию подлежит не поведение компонентов в ответ на некорректные обращения от других компонентов, а правильность обращения компонентов друг к другу. Последнее означает, фактически, проверку поведения каждого компонента (по его постусловию) только для таких обращений, которые удовлетворяют предусловию компонента. Иными словами, поскольку мы хотим убедиться в правильности реализации, а не окружения, нас не интересует поведение реализации при вызове её операций с нарушением предусловий, и такое поведение не регламентируется спецификацией. В этом смысле отношение ***ioco*** годится для тестирования только программ без предусловий, что соответствует всюду-определённости реализации по стимулам (*input enabledness*).

Семантика γ-действия предполагает, что после него любые наблюдения недостоверны. Поскольку нас интересуют только достоверные наблюдения, будем считать, что после разрушения никакие наблюдения невозможны. Разрушение считается условно-наблюдаемым действием: как событие оно может возникать при взаимодействии, но мы ограничимся только таким тестированием, при котором этого не бывает. В этом смысле разрушение аналогично ненаблюдаемому отказу. Тем самым, нас интересует только безопасное взаимодействие окружения с реализацией, не приводящее к разрушению.

Разрушение, как и τ-действия, не управляется кнопками: оно всегда разрешено. В безопасном тестировании оператор не должен нажимать кнопку, если это может привести к выполнению разрушающего внешнего действия, то есть действия, после которого реализация может разрушиться. Тем самым, γ-действие не возникает после отказа, а только после внешнего действия или в начале работы. Последний случай вырожденный: реализация конформна только такой спецификации, которая разрешает разрушение в самом начале, но

тестирование излишне и недопустимо (машину нельзя «включать»). В LTS-модели переход может быть помечен не только внешним действием или символом τ, но и символом γ. При композиции LTS добавляются два новых правила:

$s \xrightarrow{\gamma} s' \vdash st \xrightarrow{\gamma} s't;$

$t \xrightarrow{\gamma} t' \vdash st \xrightarrow{\gamma} st'.$

У нас появляется новый вид трасс – трассы, заканчивающиеся разрушением. Теперь в реализации кнопка безопасна после трассы $\sigma \in \mathbf{I}^{\downarrow}L_{\mathfrak{R}}$, если она не только не вызывает ненаблюдаемого отказа, но и не разрушающая, то есть не разрешающая разрушающего действия (условие, подчёркнутое волнистой линией):

P ***safe in*** $\mathbf{I}$ ***after*** $\sigma =_{def} (P \in \mathfrak{R} \vee \sigma\cdot\langle P\rangle \notin \mathbf{I})$ $\underset{\sim}{\&\ \forall z \in P\ \sigma\cdot\langle z,\gamma\rangle \notin \mathbf{I}}$.

Соответствующим образом модифицируются правила, которым должен отвечать протокол взаимодействия (добавляются условия, подчёркнутые волнистой линией):

$\forall \sigma \in \mathbf{\Sigma}^{\downarrow}L_{\mathfrak{R}}\ \forall R \in \mathfrak{R}\ \forall z \in L\ \forall Q \in \mathfrak{Q}$

1) R ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Leftrightarrow \underset{\sim}{\forall u \in R\ \sigma\cdot\langle u,\gamma\rangle \notin \mathbf{\Sigma}}$,

2) $\sigma\cdot\langle z\rangle \in \mathbf{\Sigma}$ $\underset{\sim}{\&\ \exists T \in \mathfrak{R}\cup\mathfrak{Q}\ z \in T\ \&\ \forall u \in T\ \sigma\cdot\langle u,\gamma\rangle \notin \mathbf{\Sigma}}$

$\Rightarrow \exists P \in \mathfrak{R}\cup\mathfrak{Q}\ z \in P\ \&\ P$ ***safe by*** $\mathbf{\Sigma}$ ***after*** σ,

3) Q ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Rightarrow \exists v \in Q\ \sigma\cdot\langle v\rangle \in \mathbf{\Sigma}$ $\underset{\sim}{\&\ \forall u \in Q\ \sigma\cdot\langle u,\gamma\rangle \notin \mathbf{\Sigma}}$.

Определение безопасной трассы реализации изменяется. Во-первых, оно опирается на модифицированное отношение ***safe in***. Во-вторых, в безопасной трассе не должно быть разрушающих $\mathfrak{R}$-отказов. Также появляется понятие безопасной трассы спецификации. Раньше в нём не было нужды, поскольку безопасность понималась только как отсутствие ненаблюдаемых отказов, а

второе правило для протокола взаимодействия гарантировало для каждого внешнего действия в трассе наличие кнопки, разрешающей это действие и не вызывающей ненаблюдаемого отказа после предшествующего префикса трассы. Теперь мы должны учитывать возможность разрушения. Безопасных трасс нет, если есть трасса $\langle\gamma\rangle$.

$$\mathit{SafeIn}(\mathbf{I}) =_{\text{def}} \{\sigma\in\mathbf{I}^{\downarrow}\mathrm{L}_{\mathfrak{R}} \mid \langle\gamma\rangle\notin\mathbf{I} \;\&\; \forall\mu\ \forall u$$

$$(\mu\cdot\langle u\rangle\le\sigma \Rightarrow \exists P\in\mathfrak{R}\cup\mathfrak{Q}\ \ P \ \textit{safe in}\ \mathbf{I}\ \textit{after}\ \mu \;\&\; (u\in P \vee u=P)\}).$$

$$\mathit{SafeBy}(\mathbf{\Sigma}) =_{\text{def}} \{\sigma\in\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}} \mid \langle\gamma\rangle\notin\mathbf{\Sigma} \;\&\; \forall\mu\ \forall u$$

$$(\mu\cdot\langle u\rangle\le\sigma \Rightarrow \exists P\in\mathfrak{R}\cup\mathfrak{Q}\ \ P \ \textit{safe by}\ \mathbf{\Sigma}\ \textit{after}\ \mu \;\&\; (u\in P \vee u=P)\}).$$

Гипотеза о безопасности теперь требует отсутствия в реализации разрушения, если оно невозможно в спецификации в той же ситуации, то есть либо с самого начала (до нажатия кнопок), либо после безопасного продолжения безопасной трассы.

$$\mathbf{I}\ \textit{safe for}\ \mathbf{\Sigma} =_{\text{def}} \quad (\langle\gamma\rangle\notin\mathbf{\Sigma} \Rightarrow \langle\gamma\rangle\notin\mathbf{I})$$

$$\&\ \forall\sigma\in\mathit{SafeBy}(\mathbf{\Sigma})\cap\mathit{SafeIn}(\mathbf{I})\ \ \forall P\in\mathfrak{R}\cup\mathfrak{Q}$$

$$(P \ \textit{safe by}\ \mathbf{\Sigma}\ \textit{after}\ \sigma \Rightarrow P \ \textit{safe in}\ \mathbf{I}\ \textit{after}\ \sigma).$$

Конформность строится не на всех, а только на безопасных $\mathfrak{R}$-трассах спецификации:

$$\mathbf{I}\ \textit{saco}\ \mathbf{\Sigma} =_{\text{def}} \ \mathbf{I}\ \textit{safe for}\ \mathbf{\Sigma}$$

$$\&\ \forall\sigma\in\mathit{SafeBy}(\mathbf{\Sigma})\cap\mathit{SafeIn}(\mathbf{I})\ \ \forall P \ \textit{safe by}\ \mathbf{\Sigma}\ \textit{after}\ \sigma$$

$$\mathit{obs}(\sigma, P, \mathbf{I})\subseteq\mathit{obs}(\sigma, P, \mathbf{\Sigma}).$$

Утверждения о рефлексивности и транзитивности конформности при отсутствии $\mathfrak{Q}$-кнопок и о существовании (в том числе, алгоритмизуемых) пополнения $\mathcal{C}$ и монотонного преобразования $\mathcal{M}$ остаются в силе и для γ-семантики (см. Главы 5 и 6). Но теперь, для γ-семантики, существует такая

пополненная спецификация, которую уже можно рассматривать не только в $\mathfrak{R}\cup\mathfrak{Q}$-семантике, но и в $\mathfrak{R}/\mathfrak{Q}$-семантике (ср. Рис.16 и Рис.17).

$\mathbf{I}\ \mathit{saco}_{\mathfrak{R}/\mathfrak{Q}}\ \mathbf{S} \Leftrightarrow \mathbf{I}\ \mathit{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ C(\mathbf{S}) \Leftrightarrow \mathbf{I}\ \mathit{saco}_{\mathfrak{R}/\mathfrak{Q}}\ C(\mathbf{S})$.

Для γ-семантики пополненная спецификация эквивалентна исходной спецификации в $\mathfrak{R}/\mathfrak{Q}$-семантике

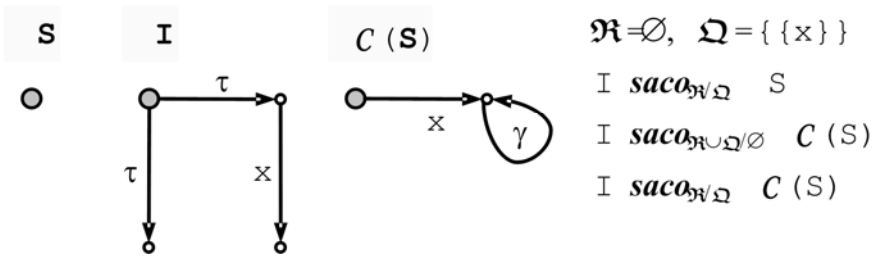


Рис.17.

Заметим, что в пополнении C вместо «демонического» продолжения трассы σ всеми допустимыми трассами вида $\sigma\cdot\langle z\rangle\cdot\lambda$, используется гамма-пополнение, продолжающее трассу σ только трассой $\sigma\cdot\langle z,\gamma\rangle$. Тем самым, не теряется информация об опасных кнопках: после демонического пополнения мы должны при тестировании проверять все добавленные трассы $\sigma\cdot\langle z\rangle\cdot\lambda$, хотя это и бессмысленно («всё разрешено»), а при гамма-пополнении добавленная трасса $\sigma\cdot\langle z\rangle$ не проверяется, поскольку ведёт к разрушению.

2.11. Дивергенция

Дивергенция (бесконечная внутренняя активность) – вовсе не обязательно ошибка «зацикливания» программы. В некоторых случаях дивергенция является правильным или неизбежным поведением. Например, тест может подменять собой не всё окружение, а только его часть. Фактически, тестируется композиция реализации с остальной частью окружения. Взаимодействие реализации с этой частью окружения может быть бесконечным, но из теста оно

видимо как бесконечная внутренняя активность, то есть дивергенция (синхронные переходы композиции – это τ-переходы). Возможна и чисто «внутренняя» дивергенция, например, «ждущий тест» в операционной системе, который продолжается бесконечно долго, пока его не прервут.

На самом деле проблема не в дивергенции самой по себе, а в выходе из неё: если внешнее воздействие имеет больший приоритет, чем внутренняя активность, дивергенция прекращается. В машине с приоритетами выполнимость τ-действий зависит от нажатой кнопки, и мы можем косвенно управлять ими и, следовательно, дивергенцией. Тогда можно говорить о *выполнимой* дивергенции: при одной нажатой кнопке (или когда нет нажатой кнопки) все τ-действия бесконечной цепочки выполнимы, а при другой – нет и, следовательно, нет «зацикливания». Выйти из дивергенции, которая начинает выполняться после кнопки “A”, можно с помощью кнопки “B”, при которой дивергенция не выполнима. Заметим, что для этого требуется переключение кнопок, то есть нажатие кнопки без наблюдения (которого может не быть). Единственный случай, когда из дивергенции нельзя гарантированно выйти, – это когда дивергенция выполнима при нажатии любой кнопки. Для машины без приоритетов такой нежелательной дивергенцией является любая дивергенция.

Дивергенция «неудобна» тем, что не позволяет получить наблюдение в ответ на тестовое воздействие за конечное время и, тем самым, продолжить тестирование после наблюдения. В этом смысле дивергенция аналогична ненаблюдаемому отказу или разрушению. Более точно: дивергенция аналогична ситуации, когда нажатие каждой кнопки может вызвать выполнение разрушающего внешнего действия или, для $\mathfrak{Q}$-кнопок, ненаблюдаемый отказ. В частности, все непустые кнопки опасны, если все внешние действия (или, по крайней мере, хотя бы одно действие из каждого непустого отказа) разрушающие. Пустая $\mathfrak{Q}$-кнопка всегда опасна, однако пустая $\mathfrak{R}$-кнопка всегда безопасна. Если бы не было пустой $\mathfrak{R}$-кнопки, мы могли бы моделировать дивергенцию, определяя все внешние действия как разрушающие. Например,

такой подход годится для γδ-семантики (δ-семантики с разрушением) βγδ-семантики (βδ-семантики с разрушением), хотя в [21,22,23,25-27] используется более грубое моделирование дивергенции не разрушающими внешними действиями, а самим разрушением. К сожалению, наличие пустой $\mathfrak{R}$-кнопки (возможность наблюдения остановки машины при запрете всех внешних действий) вынуждает нас искать другой способ моделирования дивергенции.

В машине без приоритетов поведение реализации при дивергенции предлагается моделировать специальным Δ-действием. Оно разрешается каждой кнопкой и считается условно-наблюдаемым: как событие дивергенция может возникать при взаимодействии, но мы ограничимся только таким тестированием, при котором этого не бывает. После Δ-действия никакие наблюдения невозможны (точнее, мы не можем гарантированно дождаться их за конечное время). Δ-действие очень похоже на γ-действие, и отличается только тем, что разрушение нельзя запретить, а Δ-действие запрещено, если кнопки не нажаты, но разрешается нажатием любой кнопки, что моделирует проявление дивергенции при попытке выхода из неё. Соответственно меняются отношения безопасности (добавления отмечены волнистой линией):

P ***safe in*** $\mathbf{I}$ ***after*** σ

$=_{def}$ $(P\in\mathfrak{R} \vee \sigma\cdot\langle P\rangle\notin\mathbf{I})$ & $\forall z\in P\ \sigma\cdot\langle z,\gamma\rangle\notin\mathbf{I}$ & $\sigma\cdot\langle\Delta\rangle\notin\mathbf{I}$.

$\forall\sigma\in\mathbf{\Sigma}^{\downarrow}L_{\mathfrak{R}}\ \forall R\in\mathfrak{R}\ \forall z\in L\ \forall Q\in\mathfrak{Q}$

1) R ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Leftrightarrow \forall u\in R\ \sigma\cdot\langle u,\gamma\rangle\notin\mathbf{\Sigma}$ & $\sigma\cdot\langle\Delta\rangle\notin\mathbf{\Sigma}$,

2) $\sigma\cdot\langle z\rangle\in\mathbf{\Sigma}$ & $\exists T\in\mathfrak{R}\cup\mathfrak{Q}\ z\in T$ & $\forall u\in T\ \sigma\cdot\langle u,\gamma\rangle\notin\mathbf{\Sigma}$ & $\sigma\cdot\langle\Delta\rangle\notin\mathbf{\Sigma}$

$\Rightarrow \exists P\in\mathfrak{R}\cup\mathfrak{Q}\ z\in P$ & P ***safe by*** $\mathbf{\Sigma}$ ***after*** σ,

3) Q ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Rightarrow \exists v\in Q\ \sigma\cdot\langle v\rangle\in\mathbf{\Sigma}$ & $\forall u\in Q\ \sigma\cdot\langle u,\gamma\rangle\notin\mathbf{\Sigma}$ & $\sigma\cdot\langle\Delta\rangle\notin\mathbf{\Sigma}$.

Гипотеза о безопасности теперь разрешает дивергенцию в реализации только в том случае, когда это разрешено в спецификации в той же самой ситуации, то есть после той же самой трассы, безопасной в спецификации.

При композиции может возникать дивергенция как бесконечная цепочка синхронных переходов даже в том случае, когда в операндах нет дивергенции. Это означает, что класс моделей без дивергенции не замкнут по композиции.

Иногда рассматривают тестирование, в котором возможно прямое (а не условное, как у нас) наблюдение дивергенции (Δ-наблюдение) или, так называемое, λ-наблюдение, означающее дивергенцию *или* ненаблюдаемый отказ. В обоих случаях предполагается либо возможность бесконечного наблюдения, либо ограничение сверху на время выполнения любого внешнего действия и любой конечной цепочки τ-наблюдений. Если экран пуст бесконечно долго или дольше тайм-аута после нажатия $\mathfrak{R}$-кнопки, то это означает дивергенцию, а после нажатия $\mathfrak{Q}$-кнопки – λ-наблюдение. Бесконечное наблюдение мы считаем практически нереальным и далее не рассматриваем. Если ограничение по времени используется как способ обнаружения отказа, то различие между $\mathfrak{R}$- и $\mathfrak{Q}$-кнопками стирается, и срабатывание тайм-аута в обоих случаях приходится понимать как λ-наблюдение. В дальнейшем мы будем предполагать, что дивергенция моделируется Δ-действием, а тестирование безопасно, то есть избегает ненаблюдаемых отказов, дивергенции и разрушения. Это основано на представлении о том, что при правильном взаимодействии не должно возникать тупиков, бесконечного ожидания окружением результата взаимодействия и разрушения системы.

2.12. Примеры тестовых семантик и конформностей

Рассмотрим несколько встречающихся в литературе тестовых семантик (вместе с соответствующими конформностями). Наша цель – интерпретировать

их как $\mathfrak{R}/\mathfrak{Q}$-семантики. Обзоры большинства этих семантик и конформностей можно найти у Ван Глаббека [89,90], где, правда, не рассматриваются семантики, специфические для реактивных систем, и у Тритманса [149], исследующего как раз такие семантики. Для каждой из этих семантик, интерпретируемой как $\mathfrak{R}/\mathfrak{Q}$-семантика, однозначно выделяется одна гипотеза о безопасности, поскольку в них каждое действие z, не разрешаемое $\mathfrak{R}$-кнопками, разрешается только одной $\mathfrak{Q}$-кнопкой. Все такие действия, продолжающие трассу спецификации, определяют все $\mathfrak{Q}$-кнопки, которые должны быть объявлены безопасными после этой трассы. Во всех случаях разрушение не рассматривается, а дивергенция в реализации не допускается.

Трассовая семантика (*trace semantics*) [89,90,100,144,149]. Задаётся $\mathfrak{R}/\mathfrak{Q}$-машиной с $\mathfrak{R}=\varnothing$ и $\mathfrak{Q}=\{L\}$. Трассы содержат только внешние действия. Обычно трассовый предпорядок (***tr*** – *trace preorder*) означает простую вложенность трасс. Однако мы трактуем остановку с ненаблюдаемым отказом как ошибку взаимодействия, и рассматриваем только безопасное тестирование. Гипотеза о безопасности требует: если наблюдаемая трасса в спецификации продолжается внешним действием, то она не может заканчиваться в терминальных состояниях LTS-реализации. Оператору запрещено нажимать кнопку “L” только после трассы, заканчивающейся в LTS-спецификации только в терминальных состояниях.

Семантика завершённых трасс (*completed trace semantics*) [89,90]. Задаётся $\mathfrak{R}/\mathfrak{Q}$-машиной с $\mathfrak{R}=\{L\}$ и $\mathfrak{Q}=\varnothing$. Единственный отказ L наблюдаем в терминальном состоянии реализации, когда никакое другое продолжение невозможно. Трассы содержат внешние действия и отказ L, но после него не может наблюдаться никакое действие. Любая реализация безопасна. Соответствующая конформность называется предпорядком завершённых трасс (***ct*** – *completed trace preorder*).

Семантика трасс с отказами (*failure trace semantics*) [58,89,90,111,137,149]. Задаётся ***F***-машиной с $\mathfrak{R}=\mathcal{P}(\mathtt{L})$ и $\mathfrak{Q}=\varnothing$. Трассы содержат внешние действия и произвольные отказы (***Ftraces***). Любая реализация безопасна. Соответствующая конформность называется предпорядком трасс с отказами (*failure trace preorder* или ***rf*** – *refusal preorder*). Встречается также разновидность этой семантики, когда разрешено может быть только *конечное* множество внешних действий (*refusal semantics* в [137]). Этому соответствует $\mathfrak{R}/\mathfrak{Q}$-машина, в которой $\mathfrak{R}$ – семейство всех *конечных* подмножеств алфавита $\mathtt{L}$.

Мы не останавливаемся на соответствующих эквивалентностях (*trace*, *completed trace* и *failure trace* или *refusal equivalence*), поскольку они не отвечают принципу независимости поведения реализации.

Для реактивных систем имеется ряд специфических семантик. Все они основаны на следующих ограничениях: 1) окружение не может выбирать, какую реакцию ему принимать, а какую нет: если реакции принимаются, то принимается любая реакция; 2) имеется возможность послать один стимул, не посылая других стимулов.

Семантика для реактивных систем без блокировки стимулов [148,149]. Задаётся $\mathfrak{R}/\mathfrak{Q}$-машиной с $\mathfrak{R}=\{\delta\}$, где $\delta=\{\mathtt{!y}\,|\,\mathtt{!y}\in\mathtt{L}\}$, и $\mathfrak{Q}=\{\{\mathtt{?x}\}\,|\,\mathtt{?x}\in\mathtt{L}\}$. Эту семантику будем называть δ-семантикой. Трассы содержат стимулы, реакции и единственный отказ – стационарность δ. Отношение ***ior*** (*input-output refusal relation* или *repetitive quiescence relation*), означает простую вложенность трасс $\mathbf{I}^{\downarrow}\mathtt{L}_{\mathfrak{R}}\subseteq\mathbf{\Sigma}^{\downarrow}\mathtt{L}_{\mathfrak{R}}$. Для того, чтобы тестирование было безопасным, любая трасса $\sigma\in\mathbf{I}^{\downarrow}\mathtt{L}_{\mathfrak{R}}\cap\mathbf{\Sigma}^{\downarrow}\mathtt{L}_{\mathfrak{R}}$ должна в LTS-реализации заканчиваться только в таких стабильных состояниях, в которых определены переходы по всем стимулам. Однако, если такая трасса в спецификации не продолжается стимулом $\mathtt{?x}$, то реализация заведомо

неконформна. Из-за этого спецификация, содержащая, например, только трассу ⟨?x,!y⟩ и все её префиксы, не имеет конформных реализаций.

Отмеченная неудовлетворительность отношения ***ior*** преодолевается в отношении ***ioco*** (*input-output conformance*). Это отношение опирается на гипотезу о всюду-определённости реализации по стимулам (*input enabledness*): в каждом достижимом стабильном состоянии LTS-реализации принимаются все стимулы. Однако, на самом деле, при тестировании стимул ?x не посылается в реализацию после трассы σ, которая в спецификации не продолжается стимулом ?x, то есть для конформности ***ioco*** безразлично поведение реализации после приёма такого стимула. Если **I** ***ioco*** **Σ** и σ·⟨?x⟩∈**I**, то реализация **I`**, отличающаяся от **I** только тем, что она блокирует стимул ?x после трассы σ, также безопасна (не возникает ненаблюдаемых отказов). Эти две реализации обе безопасны и не различимы при безопасном тестировании. Тем не менее, реализация **I** конформна, а реализация **I`** неконформна, поскольку не входит в домен отношения ***ioco***. Иными словами, гипотеза всюду-определённости реализации по стимулам неоправданно сильная.

Например, на Рис.18 приведена системы **A**, взаимодействие с которой происходит по «автоматному» правилу «стимул-реакция». Спецификация – LTS $\mathbf{A_0}$. Формально, она не может служить своей собственной тестируемой реализацией, так как в ней есть блокировка стимула. В то же время тесты, построенные для ***ioco***, не только не вызовут блокировку стимула (после наблюдения трассы стимул не посылается, если его нет в спецификации после этой трассы), но и не обнаружат никаких ошибок. Эти тесты вообще не отличат LTS $\mathbf{A_0}$ от тестируемой и конформной LTS $\mathbf{A_1}$.

Избыточность требования всюду-определённости реализаций для отношения ***ioco***

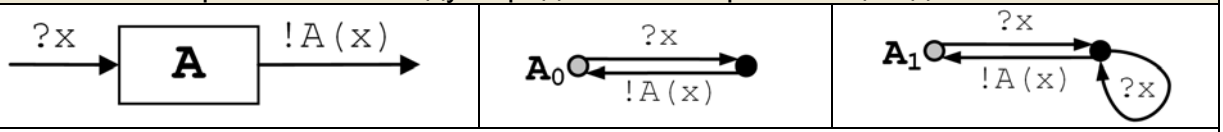


Рис.18.

Наша гипотеза о безопасности более точно выражает предусловие тестирования реализации: если трасса σ в спецификации продолжается стимулом ?x, то в реализации трасса σ не должна заканчиваться в состоянии, где стимул блокируется. Для отношения ***saco*** не может быть двух реализаций, неразличимых при безопасном тестировании, одна из которых конформна, а другая нет. В примере на Рис.18 $\mathbf{A_0}$ ***saco*** $\mathbf{A_0}$ и $\mathbf{A_1}$ ***saco*** $\mathbf{A_0}$.

Семантика для реактивных систем с блокировками стимулов. Задаётся $\mathfrak{R}/\mathfrak{Q}$-машиной с $\mathfrak{R}=\{\delta\}\cup\{\{?x\}\mid ?x\in L\}$ и $\mathfrak{Q}=\varnothing$. Эту семантику будем называть $\beta\delta$-семантикой. Трассы содержат стимулы, реакции и отказы: стационарность δ и блокировки стимулов. Любая реализация безопасна. Соответствующая конформность типа ***saco*** называется $\boldsymbol{ioco}_{\beta\delta}$ [21-23,25-27]. Именно к этой семантике нужно переходить при пополнении спецификации, заданной изначально в семантике без блокировок стимулов (δ-семантике).

Семантика для систем с мультиканалами стимулов и реакций (*MIOTS – Multi Input-Output Transition Systems*). Такие системы рассматриваются в работах Херинка и Тритманса, где предлагается конформность ***mioco*** [95,96]. Множество стимулов разбивается на подмножества – входные каналы L_I, и для каждого входного канала L_I реализация должна принимать либо любой стимул из L_I, либо ни одного. Множество реакций также разбивается на подмножества – выходные каналы L_U, но, в отличие от стимулов, не налагается ограничений на выдачу реакций реализацией. 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). Заметим, что здесь тест уже не обязан принимать все или ни одной реакции, но обязан принимать все или ни одной

реакции из каждого выходного канала. В $\mathfrak{R}/\mathfrak{Q}$-машине каждому стимулу ?x соответствует $\mathfrak{R}$-кнопка {?x}, каждому выходному каналу L_U – $\mathfrak{R}$-кнопка L_U, $\mathfrak{Q}$-кнопок нет. Указанное выше ограничение на приём стимулов является ограничением на реализацию (и спецификацию) и не затрагивает устройства машины. Любая реализация безопасна.

Отношение ***mioco*** почти совпадает с отношением ***saco*** для такой машины за одним исключением. Отношение ***mioco*** разрешает реализации 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.

Иногда в реактивных системах на окружение налагают ограничение: оно не должно «тормозить реакции», выдаваемые реализацией [115,134]. В терминах LTS это значит, что в каждом стабильном состоянии окружения/теста должны быть определены переходы по всем реакциям.

Семантика для реактивных систем без «торможения реакций» и без блокировок. Задаётся $\mathfrak{R}/\mathfrak{Q}$-машиной с $\mathfrak{R}=\{\delta\}$ и $\mathfrak{Q}=\{\delta\cup\{?x\}\mid ?x\in L\}$. В отличие от аналогичной семантики с торможением реакций гипотеза о безопасности ослабляется: если трасса σ в спецификации продолжается стимулом ?x *или реакциями*, то в реализации трасса σ не должна заканчиваться в *стационарном* состоянии, где этот стимул ?x блокируется. Посылая стимул в безопасную реализацию, мы должны быть готовы к тому, что будем наблюдать не приём стимула, а выдачу реакции. Если в реализации есть бесконечная цепочка выдачи реакций (*осцилляция* [134]), то мы не сможем послать стимул ?x с гарантированным приёмом его реализацией. Более того,

если в этой цепочке нет состояний с приёмом стимула ?x, мы об этом никогда не узнаем, непрерывно нажимая на одну и ту же кнопку “δ∪{?x}”.

Семантика для реактивных систем без «торможения реакций» и с блокировками. Задаётся $\mathfrak{R}$/$\mathfrak{Q}$-машиной с $\mathfrak{R}=\{\delta\}\cup\{\delta\cup\{?x\}\mid ?x\in L\}$ и $\mathfrak{Q}=\varnothing$. Любая реализация безопасна. Блокировка стимула наблюдается только одновременно со стационарностью (в стационарных состояниях LTS-реализации). Проблема осцилляции здесь также имеет место: мы не сможем ничего узнать о блокировках стимулов в нестационарных состояниях. Примером соответствующей конформности может служить отношение ***rioco*** [115].

Для большинства описанных выше семантик с отказами изучались разновидности, когда любой отказ или отказ определённого вида может наблюдаться только в конце трассы. После наблюдения такого отказа дальнейшие наблюдения невозможны. Это эквивалентно тому, что тестирование всё же можно продолжать после наблюдения отказа, но сам отказ «забывается», то есть не входит в трассу. В этих семантиках, кроме трасс внешних действий, рассматриваются также пары <трасса внешних действий, отказ> (*failure pairs* или, для реактивных систем с единственным отказом – стационарностью, *quiescent traces*) [70,125,126]. В $\mathfrak{R}$/$\mathfrak{Q}$-машине (как и в реактивной машине) этому могла бы соответствовать блокировка клавиатуры при наблюдении отказа. В генеративной машине Ван Глаббек предлагает считать, что переключатели нельзя перебрасывать из положения *blocked* в положение *free*, что в общем эквивалентно блокировке клавиатуры по мощности тестирования. Для семантики завершённых трасс ничего не меняется, поскольку отказ L в любом случае означает, что никаких действий больше не будет. Для остальных семантик конформности меняются:

rf→ ***te*** – тестовый предпорядок (*testing preorder*), если проверяются отказы после любых трасс [82,107,125,126,149], или отношение ***conf***, если учитываются только трассы спецификации [74,75,138,146, 147,149,153];

ior→ отношение ***iot*** (*input-output testing relation*), стационарность проверяется после любых трасс [143,149,150];

ioco→ отношение ***ioconf*** (с той же расшифровкой *input-output conformance*), учитываются только трассы спецификации [149].

Отношение ***rioco*** также допускает блокировки стимулов только в конце трассы.

На наш взгляд, все эти разновидности отношений только уменьшают мощность тестирования, но не вполне понятны причины, по которым на практике наблюдение отказа не даёт возможность продолжения тестирования с запоминанием того факта, что отказ наблюдался. Далее мы не будем на этом останавливаться.

2.13. Проблема выбора семантики тестового взаимодействия

Структура раздела:

1. Блокировка стимулов
2. Стимулы «на выбор»
3. Реакции «на выбор»
4. «Торможение» реакций

В научных кругах до сих пор не утихают споры по поводу того, на какую семантику взаимодействия следует опираться при тестировании. Поскольку разрушение обычно не рассматривается, а дивергенция в реализации не допускается, речь идёт о том, какие бывают кнопки машины тестирования и какие отказы наблюдаемы, а какие нет.

С одной стороны, имеется теория трасс с отказами, в которой для каждого подмножества действий предусмотрена своя кнопка и все отказы считаются наблюдаемыми [58,89,90,111,149]. Однако в практических тестовых системах такая семантика почти никогда не используется. На практике тестовые

возможности часто ограничены и многие отказы наблюдаться не могут. По-видимому, желанием сделать теорию более практичной можно объяснить стремление ограничиться только конечными кнопками [137].

На другом полюсе находятся исследователи, идущие от практики. Они с большим трудом признают возможность наблюдения того или иного отказа. В основном здесь речь идёт о реактивных системах, для которых все авторы признают возможность послать из теста в реализацию отдельный стимул и принимать любую реакцию от реализации. Также признаётся, что во многих (но уже не всех) практических случаях наблюдаема стационарность (отсутствие реакций). Споры идут, главным образом, о четырёх проблемах: 1) проблема блокировка стимулов, 2) проблема стимулов «на выбор», 3) проблема реакций «на выбор», 4) проблема «торможения» реакций.

Также и на практике приходится сталкиваться с разнообразием семантик тестового взаимодействия, которое строится на пересечении того, что мы можем, и того, что нам нужно. Речь идёт о тестовых возможностей: какие есть тестовые воздействия и какие есть наблюдения. То, что мы можем или не можем, – это ограничение тестовых возможностей «сверху». То, что нам требуется для проверки конформности, – это ограничение «снизу».

2.13.1. Блокировка стимулов

В ранних работах откровенно предполагалось, что никаких блокировок стимулов быть не должно: реализация в любом своём состоянии обязана принимать любой стимул, который ей посылается. В последние годы стали появляться исследования, в которых блокировки считаются наблюдаемыми (иногда частично) [21-23,25-27,115,143]. Это объясняется тем, что запрет на блокировку, как выяснилось, делает невозможным тестирование многих систем, в функциональность которых входит блокировка. Рассмотрим три примера, когда наблюдение блокировки – это ограничение «снизу».

1) Воспользуемся излюбленным примером Яна Тритманса – 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 [115] такой тип блокировки называется «физическим» (physical).

2) Математически машина «чай-кофе» – это пример FIFO-очереди (для монет) длиной один. В общем случае, если реализация – это ограниченная FIFO-очередь, то 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. То же самое имеет место вообще для сред взаимодействия ограниченной ёмкости.
3) Более практический пример – взаимодействие через пару симплексных каналов. Если связь через канал передачи стимулов разорвана, 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 [115] такой тип блокировки называется «программным» (*software*).

Запрет на блокировку стимулов в реализации иногда мотивируют тем, что блокировку 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, что во многих случаях такая мотивировка нереалистична. 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 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. Для взаимодействия через симплексные 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 появляется, например, всплывающее окно с надписью «не принято». Такой способ работы тоже встречается, но он существенно отличается от описанного, когда кнопка «бледная» и её нажать не получается.

Исторически представление о реализациях, всегда готовых принимать любые стимулы, возникло из протоколов взаимодействия по постоянно установленным каналам. Стимулы помещаются в 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.

Ян Тритманс в своей диссертации [146] 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 выполнять внутренние действия и выдавать реакции, и через некоторое время такая блокировка либо исчезает (стимул принимается), 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 этот автор больше не возвращался к проблеме блокировок стимулов, считая, что реализация всегда принимает стимулы. Исключение составляют его совместные работы с Херинком.

В работах Херинка и Тритманса по Multi Input-Output Transition Systems (MIOTS) блокировка стимулов допускается частично [95,96]. Реализация либо принимает все стимулы из данного канала, либо ни одного. В последнем случае возможно наблюдение отказа. Тем самым, каждому входному каналу соответствует свой отказ, и других отказов по стимулам нет. Здесь блокировка стимула по каналу `A` оказывается временной также и в том случае, когда реализация не имеет внутренней активности и не выдаёт реакций, но готова принять любой стимул из другого входного канала `B`, который в данный момент времени пуст (стимулы не предлагаются). Реализация ждёт, когда в канале `B` появится стимул. Если в этот канал послать стимул, работа продолжится и, возможно, в дальнейшем будет принят заблокированный стимул из канала `A`. Если каждому стимулу в MIOTS будет соответствовать ровно один канал, то все возможные отказы по стимулам – это все блокировки стимулов.

Соответствующая конформность ***mioco*** разрешает реализации блокировать стимул только в том случае, когда это возможно в спецификации. В то же время 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.

Следует 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 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 отдельных 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).

Грегори Лестиннес и Мари-Клод Годель в [115] 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: в состоянии стимул 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 после трассы с блокировками, а не о множестве блокировок после трассы без блокировок).

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 специфицирован. 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.

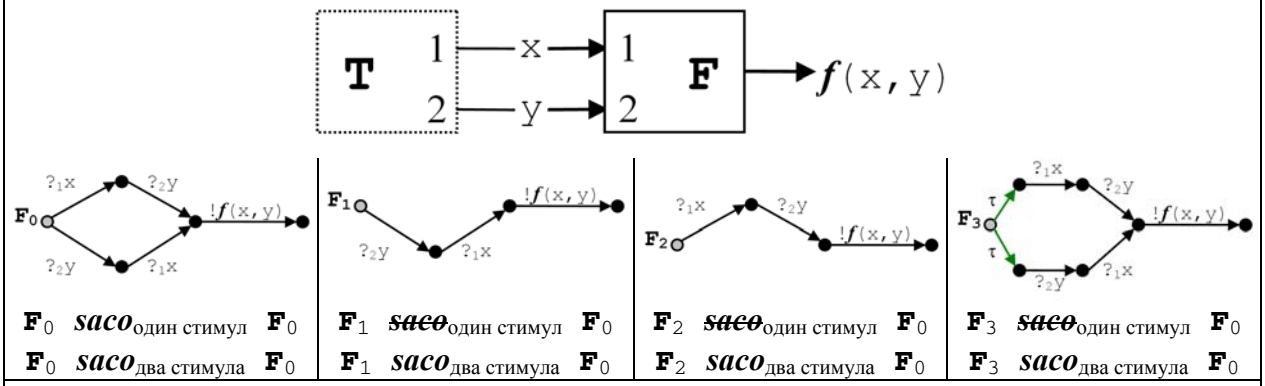
2.13.2. Стимулы «на выбор»

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 [95,96], а реализация в атомарном взаимодействии выбирает один стимул из одной очереди, тем самым не выбирая предлагаемые стимулы из других непустых очередей. Необходимость (ограничение «снизу») в передаче сразу нескольких стимулов требуется тогда, когда спецификация разрешает любой порядок приёма из портов (система **F** на Рис.19). После передачи одного стимула, тест предлагает реализации оставшиеся стимулы и так далее до тех пор, пока не будут переданы стимулы на все входные порты реализации. На рисунке показаны четыре реализации (включая спецификацию $\mathbf{F}_0$), отличающиеся порядком приёма стимулов, которые будут конформны только в том случае, если посылать из теста сразу два стимула. Этот пример представляет класс пороговых систем, которые сначала принимают в любом порядке несколько стимулов на разные входные порты, а потом выдают реакцию.

Необходимость посылки нескольких стимулов в одном тестовом воздействии: приём стимулов в любом порядке на два входных порта



$\mathbf{F}_0$ $saco_{\text{один стимул}}$ $\mathbf{F}_0$ $\mathbf{F}_1$ $saco_{\text{один стимул}}$ $\mathbf{F}_0$ $\mathbf{F}_2$ $saco_{\text{один стимул}}$ $\mathbf{F}_0$ $\mathbf{F}_3$ $saco_{\text{один стимул}}$ $\mathbf{F}_0$

$\mathbf{F}_0$ $saco_{\text{два стимула}}$ $\mathbf{F}_0$ $\mathbf{F}_1$ $saco_{\text{два стимула}}$ $\mathbf{F}_0$ $\mathbf{F}_2$ $saco_{\text{два стимула}}$ $\mathbf{F}_0$ $\mathbf{F}_3$ $saco_{\text{два стимула}}$ $\mathbf{F}_0$

Рис.19.

Необходимость во множестве стимулов возникает также 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 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). Исключением является только безопасность для случая $\mathfrak{Q}$-кнопок: может оказаться, что отдельные стимулы посылать опасно, а несколько стимулов вместе – безопасны. Если отдельные стимулы безопасны, то для наблюдения $\mathfrak{R}$-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.

2.13.3. Реакции «на выбор»

Рассмотрим проблему передачи реакций: может ли тест «выбирать», какие реакции принимать от реализации, а какие нет? Если на этот вопрос отвечают положительно, то предполагается, что в одном атомарном взаимодействии тест 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.

Семантика трасс с отказами ориентирована на такую тестовую возможность. Отношение ***mioco***, как сужение этой семантики, применяется для MIOTS, которые можно рассматривать как адекватные модели многих реальных реализаций. Такая реализация имеет несколько выходных каналов,

разбивающих алфавит реакций. Тест может принимать реакции из разных каналов независимым образом. В терминах машины тестирования, на кнопке может быть написано множество всех реакций, соответствующих одному или нескольким каналам. Тем самым, тест может обнаруживать «частичную стационарность», то есть отказ, содержащий подмножество всех реакций.

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

Рассмотрим примеры, когда приём части реакций – это ограничение «снизу». Система-приёмник **F** на Рис.19 может получать стимулы от другой системы-передатчика, в которой, наоборот, имеется несколько выходных портов и для которой передаваемые стимулы являются реакциями (система **T** на Рис.20). Чтобы обеспечить передачу всех сообщений при любом порядке их приёма, мы вынуждены (ограничение «снизу») потребовать от передатчика того же, что требуем от теста приёмника: начинать с выдачи сразу всех сообщений. Выдача двух реакций последовательно: сначала по первому порту, а потом по второму, или наоборот, является ошибкой. Если тест, как это чаще всего бывает (в частности, для ***ioco***), принимает любую реакцию, то по какому бы порту ни пришла реакция, это приходится считать правильным, и ошибка не обнаруживается. Чтобы обнаружить ошибку, нам нужно уметь в тесте принимать не все реакции, а только из одного порта: один тест обнаружит ошибку как отсутствие реакций по первому порту, а другой – по второму. На рисунке из четырёх реализаций правильна должна быть только одна: сама спецификация $\mathbf{T}_0$. Та же самая тестовая возможность требуется для тестирования широковещания: для каждого выходного порта нужно проверить, что тест, принимающий реакции из этого порта, не обнаружит их отсутствие.

Необходимость приёма части реакций в одном тестовом воздействии:
одновременная выдача двух реакций на два выходных порта

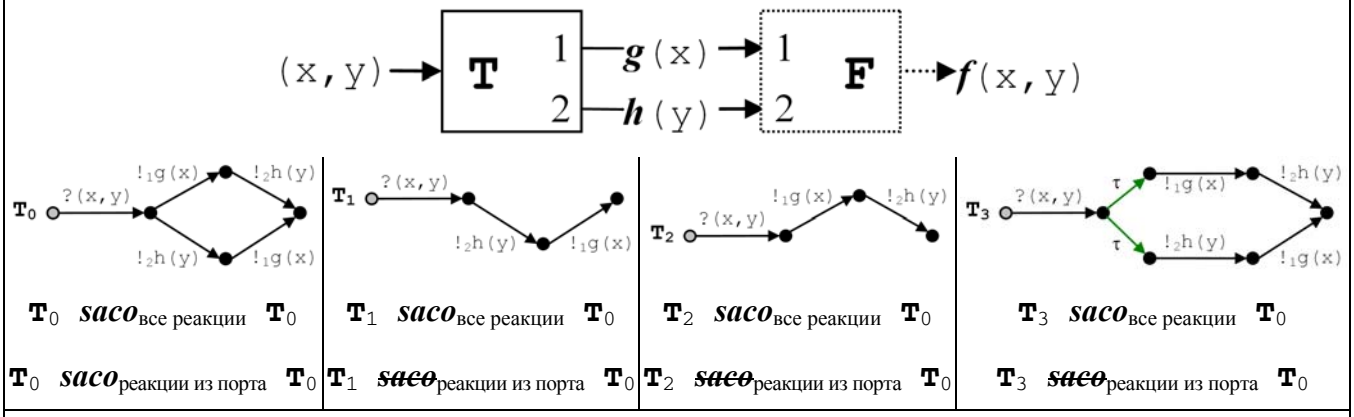


Рис.20.

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

В настоящей работе мы предполагаем, что приоритетов нет.

2.13.4. «Торможение» реакций

Итак, тестирование приоритетов между реакциями неизбежно ведёт к «торможению» реакций. Но само такое «торможение» – более общая проблема. 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 тест в некотором своём состоянии не принимать никаких реакций, а только посылать стимул?

Совмещение передачи стимулов с приёмом реакций, обычно являющееся излишним (не увеличивающим мощность тестирования), необходимо

(ограничение «снизу») для тестирования функциональности некоторых практических систем. Например, рассмотрим систему с откатами (Рис.21). В такой системе два типа стимулов: «начать транзакцию» и «откатить транзакцию», и, соответственно, два типа реакций: «транзакция завершена» и «откат выполнен». После начала транзакции реализация часто выполняет её в два этапа так, что откат возможен только на первом этапе, а на втором этапе стимул «откатить» отвергается, то есть блокируется. Если блокировки ненаблюдаемы (ограничение «сверху»), мы не можем давать команду «откатить» и, тем самым, не можем тестировать интересующие нас реализации. Однако мы можем обойти это ограничение «сверху», если одновременно с посылкой стимула «откатить» будем принимать реакции. Отказа не будет, так как всегда либо будет принят откат, либо выдана реакция «транзакция завершена».

Необходимость совмещение посылки стимула с приёмом реакций: системы с откатом

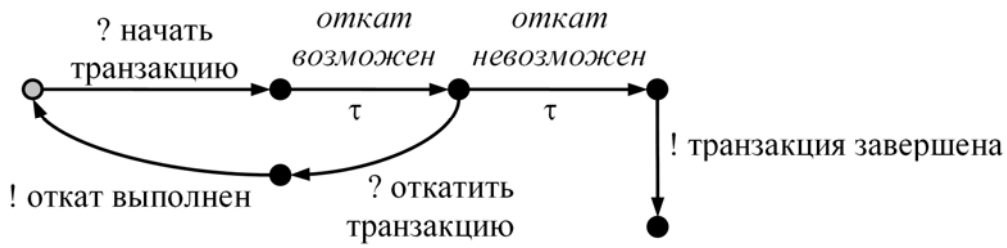


Рис.21.

Теперь обсудим допустимость торможения реакций. Но сначала посмотрим, какие выводы следует сделать из запрета на торможение. Иными словами, как такое ограничение тестовых возможностей влияет на конформность.

В стационарном состоянии, где реализация не выдаёт никаких реакций и только ждёт стимулов, никакого торможения реакций не будет даже тогда, когда тест никаких реакций не принимает – их всё равно нет. Стационарность обнаруживается как θ-наблюдение при приёме всех реакций. После этого тест 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 тест может посылать другой стимул.

Другая картина складывается в нестационарном состоянии s реализации, когда тест посылает стимул и одновременно принимает все реакции. Тест 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 могла бы вычисляться не по θ-наблюдению, а как наблюдение-отрицание, означающее, что среди всех возможных наблюдений нет наблюдения приёма стимула. Такое наблюдение-отрицание мы рассмотрим ниже.

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 $\boldsymbol{ioco}_{\beta\delta}$ для таких βδ-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 [115]). 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, когда мы хотим наблюдать блокировки в нестационарных стабильных состояниях для того, чтобы учитывать их в конформности.

Теперь обсудим допустимость торможения реакций.

Идея о том, что тест 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.

Тест посылает стимулы не в реализацию, а во входную очередь, и потому тест «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, тест принимает реакции из выходной очереди, а не из реализации непосредственно. Если очередь неограниченная, то тест, 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 тест 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 теста 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) запрещается делать выбор между реакциями одного канала, но не между каналами, то есть те или иные каналы могут тормозиться тестом. В модели взаимодействующих автоматов – (*Communicating Finite State Machines – CFSM*) [73] 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 [102]. 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 теста 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 теста 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 тест 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 тест 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 тест. 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 тестом), 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 тест, а тест 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 теста 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 SEND+WAIT). Задержка критична только при выдаче реакций, когда возврат управления из примитива SEND или SEND+WAIT 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. Например, для выдачи реакций может устанавливаться тайм-аут, и тогда из примитива SEND мы можем вернуться с двумя кодами ответа: «реакция выдана» и «истёк тайм-аут». 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. Тест 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 реализации или 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 (по коду ответа «истёк тайм-аут» в примитиве SEND) 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 [10], 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 тест посылает стимул, не принимаемый реализацией, можно было бы понимать как переход по стимулу с этим последующим альтернативным поведением. Однако остаётся случай, когда в состоянии реализации определены переходы по всем стимулам, но альтернативное поведение всё равно существует в случае, когда тест 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.

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.

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

Петренко, Евтушенко и Хуо в [134] предложили интересное решение: разделить тест 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 [10] 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 [134] 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 [10], 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 взаимодействия. В этом случае мы не можем производить синхронное тестирование и вынуждены ограничиваться более грубым асинхронным тестированием. На наш взгляд, конформность 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 взаимодействия, не тормозящей реакции реализации). Здесь, однако, возникает проблема сохранения конформности, которую мы обсудили выше.

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 конформности становятся несколько слабее. Примером служат ***rioco*** и ***queued-quiescent trace-included***.

Резюме. Общий вывод, который автор делает из анализа этой дискуссии, состоит в следующем. Вряд ли имеет смысл говорить о том, что одни семантики взаимодействия «лучше» или «хуже» других. В различных практических случаях приходится выбирать ту семантику и, соответственно, ту конформность, которые наиболее точно соответствуют имеющимся тестовым возможностям и пониманию «правильности» программы. Проблема лишь в том, что эти семантики и конформности нуждаются в обобщающей теории с единой системой понятий, единой методологией и даже едиными алгоритмами генерации тестов, пополнения и монотонного преобразования спецификаций. В качестве такой теории предлагается теория $\mathfrak{R}/\mathfrak{Q}$-семантики и безопасная конформность ***saco***, параметризуемая этой семантикой, то есть выбором алфавита внешних действий и семейств $\mathfrak{R}$ и $\mathfrak{Q}$. Разумеется, этим не исчерпывается всё многообразие семантик взаимодействия, но всё же покрывается достаточно широкий и практически важный класс семантик, основанных только на наблюдаемом поведении (не на соответствии состояний) и дополнительно учитывающих разрушение и дивергенцию.

2.14. Трассы готовности

При тестировании могут быть дополнительные тестовые возможности, которые в машине задаются с помощью, так называемых, *лампочек меню*, соответствующих внешним действиям (Рис.22). Лампочка для действия `z` горит, когда действие `z` определено в реализации (она могла бы выполнить это действие, если бы оно было разрешено). Если машина активна, состояние лампочек говорит о действиях, определённых, быть может, не в текущем, а в только что пройденных состояниях. Поэтому обычно считается, что состояние лампочек достоверно только тогда, когда машина стоит в стабильном состоянии `s`. Лампочки меню позволяют в момент остановки машины определить *множество готовности* (*ready set*)– множество всех внешних действий, определённых в состоянии `s`, то есть готовых к выполнению. Это

множество обозначим через ***ready***(s). Трассы, в которых, кроме внешних действий, могут встречаться множества готовности, называются трассами готовности (*ready traces*), а соответствующие семантика, предпорядок или эквивалентность – семантикой, предпорядком или эквивалентностью трасс готовности (*ready trace semantics, ready trace preorder or equivalence*) [59,139,140].

Рис.22.

Трассы готовности обладают свойством *генеративности*: по трассе готовности мы можем получить все трассы отказов, которые могли бы наблюдаться при выполнении реализацией той же самой цепочки переходов. Иными словами, множество трасс готовности однозначно определяет множество трасс с отказами. Однако без лампочек меню мы не можем, наблюдая трассу с отказами, определить, является ли она дополнением (заменой отказов на их дополнения до алфавита) трассы готовности или нет. При отказе нам известно, что все разрешённые действия не определены в реализации. Однако про запрещённые действия мы не знаем, определены они или нет. Кроме того, в параметризованной машине, вообще говоря, не все дополнения множеств готовности являются наблюдаемыми отказами (соответствуют $\mathfrak{R}$-кнопкам).

Как и для семантик с отказами, встречается модификация, когда наблюдение готовности не позволяет продолжать тестирование. Тогда говорят о парах <трасса внешних действий, множество готовности> (*ready pairs*), семантике, предпорядке или эквивалентности готовности (*readiness semantics, ready preorder or equivalence*) [130]. Аналогично машине тестирования с конечными отказами, рассматривается машина, в которой можно узнать только

о *конечных* множествах готовых к выполнению действий. Ван Глаббек предлагает понимать это так, что лампочка меню – это одновременно кнопка, и может гореть только, если её нажал оператор. Нажимая по одной лампочки-кнопки после остановки машины в стабильном состоянии (зелёная лампочка погасла), оператор узнаёт, определены или нет соответствующие действия. Однако, если алфавит действий бесконечен, не известно, является ли полученное множество определённых действий множеством готовности, то есть множеством *всех* определённых действий, или нет.

Тестовые возможности, моделируемые лампочками меню, конечно, увеличивают мощность тестирования, но их практичность вызывает сомнения. Наблюдаемость отказа означает, что окружение может узнать, выполнилось или нет какое-либо действие из тех, что оно потребовало от реализации, и, если выполнилось, то какое именно, и использовать это знание для коррекции дальнейшего взаимодействия. Иными словами, оно рассчитывает не только на уведомление о выполнении запрошенного действия, но и на уведомление об отказе в выполнении. Это довольно естественное поведение. Однако для определения множества готовности, окружение должно узнать, какие действия реализация в принципе *могла бы* выполнить в данном стабильном состоянии. Для этого, по-видимому, требуется специальная операция опроса, которая далеко не всегда имеется в интерфейсе реализации.

Правда, бывают и исключения: например, в графических интерфейсах такое меню определённых действий может появляться на экране, иногда в виде списка всех (или части) действий, в котором действиям, которые не могут выполняться, соответствуют «бледные кнопки». Но даже в этом примере действия в графическом меню – это только стимулы, а какие реакции реализация выдаст в ответ на нажатие той или иной кнопки из графического меню, вообще говоря, неизвестно. Иными словами, для этого примера нужно рассматривать машину тестирования, в которой лампочки меню имеются не для всех, а лишь для некоторых действий. Можно предложить более общую тестовую возможность *частичных* множеств готовности: в стабильном

состоянии система сообщает статус каждого действия: 1) определено – лампочка горит зелёным цветом, 2) не определено – лампочка горит красным цветом, или 3) не известно – лампочка не горит. Далее мы будем рассматривать полные множества готовности, когда третьего случая нет.

Теперь рассмотрим вопрос о том, как окружение может использовать информацию о множествах готовности. Наблюдение готовности происходит, когда машина стоит в стабильном состоянии. Это даёт окружению возможность вычислять любые отказы. Поэтому окружение может разрешать только такие множества действий, хотя бы одно из которых реализация может выполнить. При таком протоколе взаимодействия ненаблюдаемые отказы не возникают после того, как машина остановилась, хотя по-прежнему могут возникать, если кнопка нажимается после внешнего действия (или в начале работы). Отказы никаких дополнительных наблюдений не дают, поэтому и не рассматриваются смешанные трассы, в которых были бы и множества готовности и отказы. Естественно, мы по-прежнему должны уметь обнаруживать остановку машины, что делается с помощью θ-наблюдения в $\mathfrak{R}/\mathfrak{Q}$-машине после нажатия $\mathfrak{R}$-кнопки или с помощью зелёной лампочки в генеративной и реактивной машинах. При остановке в трассу помещается наблюдаемое множество готовности. В $\mathfrak{R}/\mathfrak{Q}$-машине гипотеза о безопасности ослабляется: она касается поведения реализации только после таких трасс, которые не заканчиваются на множество готовности (пустая трасса и трасса, заканчивающаяся внешним действием). После такой трассы σ $\mathfrak{Q}$-кнопка “P” безопасна в спецификации $\mathbf{\Sigma}$, если она безопасна после любого продолжения этой трассы множеством готовности R, которое возможно в спецификации: $\forall R\ \sigma\cdot\langle R\rangle\in\mathbf{\Sigma} \Rightarrow R\cap P\neq\varnothing$.

В отличие от внешнего действия и отказа для множества готовности естественно считать, что конформность означает не равенство $R_i=R_s$ реализационного множества готовности R_i некоторому спецификационному множеству готовности R_s после той же самой трассы, а обратную вложенность

$R_i \supseteq R_s$. Иными словами, если реализация после трассы может оказаться в состоянии, где определено некоторое множество действий R_i, то это должно быть разрешено спецификаций в том смысле, что в спецификации после этой трассы должно быть состояние, в котором определено не большее множество действий R_s. Это похоже на требование *обязательного* поведения: реализация не должна предлагать «меню» действий *меньше*, чем это разрешает спецификация.

Будем говорить, что трасса реализации *мажорируется* трассой спецификации, если они имеют одну длину и в соответствующих позициях стоят либо одинаковые внешние действия, либо вложенные множества готовности. При наличии разрушения нужно учитывать только неразрушающие внешние действия спецификации (разрушающие действия спецификации не проверяются и поэтому в реализации могут быть не определены). Для множеств трасс готовности по-прежнему применяется разрешительный принцип, но только трасса реализации разрешена не тогда, когда она сама есть во множестве разрешённых трасс спецификации, а когда там есть мажорирующая её трасса.

Такое отношение можно назвать безопасной конформностью с трассами готовности и обозначить как ***resaco***. В $\mathfrak{R}/\mathfrak{Q}$-семантике для LTS оно связано с отношением ***saco*** через некоторое монотонное пополнение $\mathcal{MC}$:

$\mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}\ \mathbf{S} \Leftrightarrow \mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathcal{MC}(\mathbf{S}) \Leftrightarrow \mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}\ \mathcal{MC}(\mathbf{S})$

$\Leftrightarrow \mathbf{I}\ \boldsymbol{resaco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathcal{MC}(\mathbf{S}) \Leftrightarrow \mathbf{I}\ \boldsymbol{resaco}_{\mathfrak{R}/\mathfrak{Q}}\ \mathcal{MC}(\mathbf{S})$.

Это утверждение пока что является гипотезой. Идея его доказательства могла бы заключаться в том, чтобы в качестве преобразования $\mathcal{MC}$ взять объединение конформных реализаций. Заметим, что эквивалентность $\mathbf{I}\ \boldsymbol{resaco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathbf{S`} \Leftrightarrow \mathbf{I}\ \boldsymbol{resaco}_{\mathfrak{R}/\mathfrak{Q}}\ \mathbf{S`}$ верна только для спецификации $\mathbf{S`} = \mathcal{MC}(\mathbf{S})$, а не для произвольной спецификации $\mathbf{S`}$.

Множества готовности обладают важным свойством *вычислимости* при композиции LTS: стабильность и множество готовности композиционного

состояния st однозначно определяются стабильностью и множествами готовности состояний-операндов s и t. При композиции LTS в алфавитах L и M имеем:

st ***стабильно*** $\Leftrightarrow$ s ***стабильно*** & ***ready***(s)$\cap$***ready***(t)=$\varnothing$

& t ***стабильно*** & ***ready***(s)$\cap$***ready***(t)=$\varnothing$,

st ***стабильно*** : ***ready***(st)=(***ready***(s)\M)$\cup$(***ready***(t)\L).

Это даёт возможность определить операцию композиции трасс готовности, которая по паре трасс λ и μ возвращает множество трасс. Это множество состоит из всех трасс, в которых подтрасса асинхронных действий из $\{\gamma\}\cup L\setminus M$ ($\{\gamma\}\cup M\setminus L$) совпадает с подтрассой таких же действий трассы λ (μ), а каждое множество готовности является композицией соответствующих множеств готовности в трассах-операндах. Операция ***ReadyTraces*** взятия множества трасс готовности LTS оказывается *аддитивной* относительно операций композиции LTS и трасс готовности:

ReadyTraces(**S**$\|$**T**) =$\cup$(***ReadyTraces***(**S**)$\|$***ReadyTraces***(**T**)).

Если дивергенция моделируется Δ-действием, мы должны также компоновать бесконечные трассы готовности: если трассы-операнды имеют бесконечные постфиксы противоположных действий, то композиционная трасса заканчивается Δ-действием.

Генеративность и аддитивность трасс готовности, как кажется, позволяют построить замкнутую трассовую теорию, включающую как конформность (даже если она основано на трассах с отказами, а не на трассах готовности), так и композицию трассовых моделей. Именно это, на наш взгляд, делает трассы готовности полезными в теории конформности. Тем не менее, в этой работе мы не рассматриваем тестовую возможность, моделируемую лампочками меню, и трассы готовности. Вместо этого в главе 6 будут введены, так называемые, ϕ-трассы, обладающие теми же свойствами генеративности и аддитивности. Теория ϕ-трасс используется для решения проблемы монотонности.

2.15. Репликация и симуляции

Наблюдаемое поведение тестируемой системы зависит от того, когда и какие кнопки нажимает оператор. Для того, чтобы смоделировать возможные варианты поведения оператора, в машине тестирования используется, так называемая, *кнопка репликации*. Она позволяет создать несколько копий машины в данный момент времени и продолжить работу с каждой копией независимо. В параметризованной машине мы предполагаем, что такая репликация выполняется перед началом работы (Рис.23). В этом случае различные копии машины имитируют различные «сеансы» тестирования: после каждого сеанса машину можно заставить работать с начала для следующего сеанса. Репликация отличается от такой сеансовой работы только тем, что абстрагируется от числа возможных сеансов: оно может быть бесконечным. Такая репликация необходима для потенциальной возможности получить все конечные трассы (в машине с приоритетами – истории) в конечных тестовых экспериментах.

Репликация при включении машины

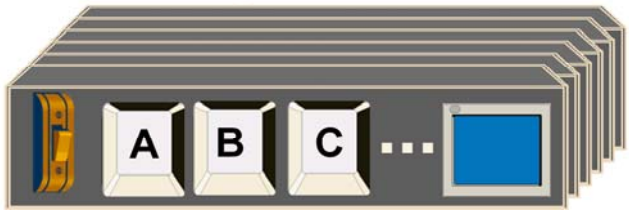


Рис.23.

В машинах Ван Глаббека и Милнера используется гораздо более сильная репликация в любой момент времени (Рис.24). С помощью такой репликации мы можем получить информацию о состоянии машины, в котором она оказалась после трассы σ, в виде множества трасс, наблюдающихся в этом состоянии. (Тем самым, состояния различаются с точностью до этого множества трасс). Если в каждой копии после наблюдения трассы σ сделана репликация, то в `i`-ой копии мы получаем множество продолжающих трасс Σ_i,

а суммарно имеем семейство $\Sigma=\{\Sigma_i \mid i=1,2,\ldots\}$. Если репликация делается только перед началом работы (только сеансы тестирования), то мы можем узнать лишь объединение $\cup\Sigma$, то есть всё множество трасс, которыми продолжается трасса σ во *всех* состояниях после этой трассы. Какие их этих трасс к каким состояниям машины относятся, то есть каковы «слагаемые» $\Sigma_1, \Sigma_2, \ldots$, нам неизвестно.

Репликация в любой момент времени

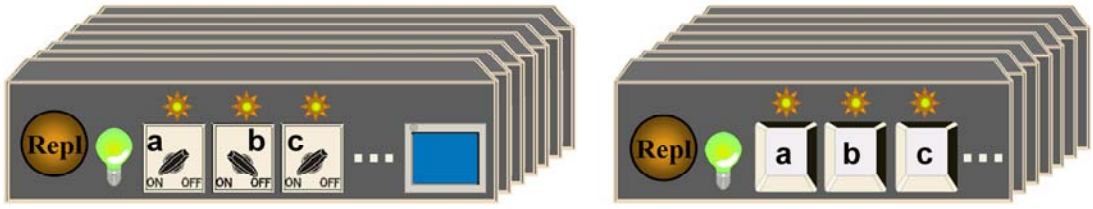


Рис.24.

Именно репликация в любой момент времени позволяет вводить такие конформности, как симуляции (*simulations*) и бисимуляции (*bisimulations*) – отношения, учитывающие так или иначе состояния системы. К таким отношениям относятся строгая (*strong*) и слабая (*weak*) симуляции [131,122], симуляция задержек (*delay simulation*) [54,121,152], симуляция ветвления (*branching simulation*) [88,127] и другие. Ван Глаббек рассматривает также различные модификации репликации, зависящие от разного рода ограничений на состояние машины, в котором можно делать репликацию (например, только в стабильных состояниях), и от числа копий машины, которые можно создавать при однократном нажатии на кнопку репликации. Эти модификации определяют соответствующие модификации отношений симуляции и бисимуляции. Для связи трассы σ, наблюдаемой до репликации, с её продолжениями после репликации, вводятся специальные операции типа «конъюнкции». Для репликации перед началом работы такие операции излишни, поскольку трасса σ всегда пуста. Репликация в любой момент времени, на наш взгляд, может считаться практически значимой только в редких специальных случаях, и далее мы её не рассматриваем.

2.16. Глобальное тестирование

Наблюдаемая трасса получается, вообще говоря, недетерминированным способом, то есть она не однозначно определяется последовательностью нажатий кнопок оператором. При одной и той же последовательности кнопок наблюдаемая трасса определяется тем, как машина выполняет на каждом шаге недетерминированный выбор одного из нескольких выполнимых действий. Недетерминизм выбора действия можно понимать как результат абстрагирования от неких не учитываемых внешних факторов – погодных условий, которые определяют этот выбор детерминировано.

Заметим, что для машины с приоритетами следует не только запоминать истории вместо трасс, но также дополнительно учитывать временные задержки, которые делает оператор между наблюдением и последующим нажатием кнопки или между двумя нажатиями кнопок при их переключении без наблюдения. Поэтому мы включаем в погодные условия также те факторы, которые влияют на «свободу воли» оператора, определяя те или иные временные задержки при нажатии кнопок. Это вполне естественно, если учесть, что оператор моделирует тест, который является программой компьютера. Такая программа недетерминирована только на некотором уровне абстракции, когда мы отвлекаемся от других программ или аппаратуры, влияющих на её поведение.

Если машина не имеет приоритетов, нам достаточно считать, что оператор работает достаточно быстро. Погодные условия будут включать только факторы, влияющие на работу машины, а не оператора. Это, конечно, не означает, что в некоторых тестовых экспериментах оператор не может работать медленно. Это означает лишь, что любая трасса, которая может быть получена при медленной работе оператора, может быть получена при быстрой работе.

Для полноты тестирования мы должны предполагать, что любые погодные условия могут быть воспроизведены в эксперименте. В формализме машины тестирования это можно понимать так, что при репликации создаётся

достаточное число копий для каждого варианта погодных условий. Если такая возможность есть, тестирование называется *глобальным* [121]. Заметим, что мы абстрагируемся от количества вариантов погодных условий. Здесь нам важна только потенциальная возможность проверить поведение системы при любых погодных условиях и любом поведении оператора.

При репликации должна быть создана, по крайней мере, одна копия для каждого сочетания теста как инструкции оператору и варианта погодных условий. Поскольку заранее известно, какой тест прогоняется на машине, мы можем считать что копии промаркированы тестами, тем более, что при тестировании нас интересуют не вообще все возможные тесты, а лишь тесты некоторого полного набора. Таким образом для каждого теста из набора создаётся столько копий, сколько есть вариантов погодных условий. Копия с некоторым вариантом погодных условий для данного модельного теста моделирует прогон реального теста на тестируемой системе.

Конечно, на практике используются только конечные тесты, а также должны быть конечными число тестов в тестовом наборе и число прогонов каждого теста. Поскольку тесты конечные, полный набор, как правило, содержит бесконечное число тестов. Кроме того, без дополнительных условий мы не можем быть уверены, что провели тестовые испытания каждого теста для всех возможных погодных условий. Возможны различные решения этих проблем.

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. Для конечного набора тестов предполагают, что, если реализация ведёт себя правильно на этих тестах, то она будет вести себя правильно на всех тестах полного набора. Для конечного числа прогонов теста предполагают, что, если реализация ведёт себя правильно при некоторых погодных условиях, то она будет вести себя правильно при любых погодных условиях [15].

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 [69].

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 [86,120].

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 [133]. Обоснованием может служить то, что во многих случаях заранее известно, что интересующие нас реализации детерминированы.

Вместо полного, но бесконечного, набора тестов на практике приходится использовать конечные наборы, которые только значимы: если тест выносит вердикт ***fail***, то реализация не конформна. Такой конечный набор строится по тому или иному *критерию покрытия*, чтобы покрыть все интересующие нас классы ситуаций (ошибок) [91,93,154]. Теоретически конечный набор можно получить фильтрацией по критерию покрытия перечислимого полного набора, хотя на практике обычно используются более прямые методы построения нужного набора. Достаточно общий подход сводится к тому, что вместо исходной спецификационной модели используется более грубая, так называемая, тестовая модель. Тестовая модель – это результат факторизации

исходной LTS-спецификации по отношению эквивалентности переходов, что обычно сводится к эквивалентности состояний и/или действий [5]. Иногда при факторизации недетерминизм исчезает. Разумеется, чтобы такой подход был оправданным, нужны мотивированные реализационные гипотезы о том, что все ошибки, которые возможны в реализации, обнаруживаются при тестировании по факторизованной спецификации (вообще по конечному набору, удовлетворяющему критерию покрытия).

Примером практического тестирования может служить тестирование конечного автомата по спецификации, заданной также в виде конечного автомата [11,12,15,56,112,113]. Если у нас есть специальная операция, позволяющая достоверно и напрямую опросить текущее состояние реализации (*status message*), то, как известно, полное тестирование сводится к обходу графа переходов автомата и применению операции опроса в каждом проходимом состоянии [11,15-18,84,113]. Обход графа переходов используется также в случае тестирования методом «чёрного ящика», когда состояния реализации не видны. Но здесь для полноты тестирования требуются реализационные гипотезы, компенсирующие отсутствие тестовой возможности достоверного опроса состояний [11,12,15]. Этот подход переносится и на общий случай LTS для реактивных систем [13]. В частности, когда используется, так называемое, стационарное тестирование, при котором стимулы подаются в реализацию только в её стационарных состояниях (в этом случае также снимается проблема торможения реакций) [10].

2.17. Бесконечные и отрицательные наблюдения

Бесконечное наблюдение – это возможность проводить тестовый эксперимент бесконечно долго и получать бесконечную трассу. Для конформностей, основанных на конечных трассах, бесконечная трасса не добавляет ничего, чего не давало бы множество всех её конечных префиксов. Исключение составляет прямое наблюдение дивергенции и λ-наблюдение, о которых речь шла выше. Бесконечные наблюдения и конформности,

основанные на бесконечных трассах, обычно не рассматриваются как значимые для практического тестирования.

Отрицание $\neg\sigma$ означает, что трасса σ отсутствует во множестве всех наблюдаемых трасс, которые мы можем гарантированно получить с помощью данной машины тестирования для проверки данной конформности. Отрицательное наблюдение предполагает, что мы можем (хотя бы потенциально) получить все наблюдаемые трассы. Для этого требуется репликация (хотя бы перед началом работы) и глобальное тестирование. Фактически, отрицательное наблюдение – это наблюдение, «вычисляемое» по положительным, то есть «реальным» наблюдениям. При репликации только перед началом работы всё множество наблюдаемых трасс представляет собой объединение множеств трасс для каждого теста.

Если бы мы не запрещали тупик во взаимодействии окружения и реализации, то при отсутствии дивергенции мы могли бы вычислять некоторые отказы как отрицательные наблюдения. Действительно, пусть трасса σ не продолжается ни одним действием из $P\in\mathfrak{R}\cup\mathfrak{Q}$ во множестве наблюдаемых трасс, то есть имеет место $\neg\sigma\cdot\langle z\rangle$ для каждого $z\in P$. Тогда трасса σ в реализации продолжается либо дивергенцией, либо отказом P, то есть при отсутствии дивергенции можно вычислить трассу $\sigma\cdot\langle P\rangle$. Однако отсюда не следует, что можно наблюдать все отказы: трасса σ может продолжаться в реализации как отказом P, так и некоторыми действиями $z\in P$. Для наблюдения всех отказов нужна уже репликация в любой момент времени. Кроме того, возникает проблема с продолжением после отказа. Для того, чтобы, наблюдая трассу $\sigma\cdot\mu$, узнать, есть или нет трасса $\sigma\cdot\langle P\rangle\cdot\mu$, мы должны определить, начинается ли трасса μ в стабильном состоянии после трассы σ. Если трасса σ заканчивается некоторым отказом R, то это так, но как получить сам этот отказ R? В целом получается, что, даже в том случае, когда разрешены тупики и отрицательные наблюдения отказов, нам нужен какой-то

реальный способ наблюдения остановки машины: зелёная лампочка или $\mathfrak{R}$-отказ в $\mathfrak{R}/\mathfrak{Q}$-машине.

Аналогичная ситуация складывается с вычислением множеств готовности. Без репликации в любой момент времени мы можем вычислить лишь множество действий, продолжающих трассу, то есть объединение множеств готовности в стабильных состояниях плюс множество действий, определённые в нестабильных состояниях после трассы. Для определения слагаемых (множества готовности в состояниях после трассы) нужна репликация в любой момент времени и реальное наблюдение остановки машины.

Если мы запрещаем при правильном взаимодействии тупик, дивергенцию и разрушение, то отрицательные наблюдения ничего не дают для конформностей, основанных на принципе независимости: поведение реализации правильно или неправильно *независимо* от других её поведений. Окончательный вердикт равен конъюнкции вердиктов во всех прогонах всех тестов из полного набора.

2.18. Приоритеты

О преимуществах приоритетов в моделях и проблемах тестирования реализаций с приоритетами выше было сказано уже достаточно. К сожалению, на сегодняшний день в теории тестирования не предусматривается никакого встроенного механизма приоритетов действий по отношению друг к другу. В частности, на этом построена классификация Ван Глаббека, на что ему в устном сообщении указал Ян Бергстра [90]. В данной работе также предполагается отсутствие приоритетов.

2.19. Выводы

Мы рассмотрели различные тестовые возможности, задаваемые с помощью машины тестирования. Среди них выделяется набор теоретически достаточно мощных и практически значимых возможностей, определяющих

$\mathfrak{R}/\mathfrak{Q}$-машину. Эта машина задаёт $\mathfrak{R}/\mathfrak{Q}$-семантику тестирования, которая строится на наблюдениях внешних действий и отказов.

Нововведениями являются:

1) Метод формализации тестового эксперимента с помощью параметризуемой машины тестирования. Целью формализации является определение семантики взаимодействия. Отличие от реактивной машины Милнера и генеративной машины Ван Глаббека заключается в том, что вместо нажатия одной или нескольких кнопок, каждая из которых соответствует одному разрешаемому действию, предлагается нажимать одну кнопку, которой соответствует множество таких действий. Кроме того, для каждой кнопки указывается, соответствует ли её «кнопочному» множеству наблюдаемый отказ или нет. Набор таких кнопок как раз и является параметром машины, с помощью которого задаётся та или иная семантика. Параметризация семействами наблюдаемых и ненаблюдаемых отказов позволяет учитывать те или иные ограничения на (правильное) взаимодействие. Также в отличие машин Милнера и Ван Глаббека репликация выполняется только перед началом работы, что соответствует конформностям, основанным только на семантике наблюдаемого поведения, но не на соответствии состояний. Кроме всех таких семантик, задаваемых машинами Милнера и Ван Глаббека, появляется возможность единообразно определять многие другие, применяемые в теории и на практике, семантики, в частности, семантику отношения ***ioco***.
2) Разрушение как запрещённое действие, которое возможно, но не должно выполняться при правильном взаимодействии.
3) Моделирование дивергенции Δ-действием, которое при правильном взаимодействии возможно, но только в конце его (без продолжения).

Предлагаются основанные на такой семантике понятие безопасного тестирования, реализационная гипотеза о безопасности и безопасная конформность (***saco***), отвечающая принципу независимости наблюдений. Эти

понятия вводятся неформально, дальнейшие главы посвящены формальному их рассмотрению.

Также мы сформулировали ряд утверждений о связи отношений ***saco*** в $\mathfrak{R}/\mathfrak{Q}$- и $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантиках. Переход к семантике без $\mathfrak{Q}$-отказов осуществляется с помощью преобразования пополнения, решающего проблему рефлексивности конформности, и монотонного преобразования, решающего проблему монотонности (сохранение при композиции) конформности. В [27] эти утверждения доказаны для реактивных систем, в которых наблюдаемыми отказами могут быть только стационарность и блокировки стимулов. Для общего случая эти вопросы формально рассматриваются в следующих главах.

Глава 3. Модель наблюдаемых трасс

Структура главы:

1. Определение модели
2. Полная модель (*F*-модель)
3. Объединение и пересечение моделей
4. Машина тестирования и трассовая модель
5. Разложение трассовой модели на поддеревья
6. Безопасность и конформность
7. Генерация тестов
8. Выводы

В этой главе мы рассмотрим трассовую модель как множество наблюдаемых трасс $\mathfrak{R}/\mathfrak{Q}$-машины тестирования. Мы дадим интенсиональное определение $\mathfrak{R}$-модели как множества $\mathfrak{R}$-трасс с определёнными свойствами.

Кроме того, мы определим трассовую ***F***-модель, соответствующую множеству наблюдаемых трасс ***F***-машины. Мы покажем, что $\mathfrak{R}$-проекция ***F***-модели (подмножество её $\mathfrak{R}$-трасс) является $\mathfrak{R}$-моделью, и любую $\mathfrak{R}$-модель можно расширить до ***F***-модели, $\mathfrak{R}$-проекцией которой она будет являться. Можно считать, что ***F***-модель описывает, как устроена система «на самом деле», а её $\mathfrak{R}$-проекция – это «взгляд» на систему, определяемый тестовыми возможностями по управлению и наблюдению, описываемыми $\mathfrak{R}/\mathfrak{Q}$-машиной.

Для дальнейшего нам также понадобится утверждение о том, что объединение любого множества $\mathfrak{R}$-моделей является $\mathfrak{R}$-моделью, а пересечение этим свойством не обладает.

После этого покажем, что множество наблюдаемых трасс $\mathfrak{R}/\mathfrak{Q}$-машины тестирования является $\mathfrak{R}$-моделью. Обратное утверждение (любая $\mathfrak{R}$-модель является множеством трасс некоторой $\mathfrak{R}/\mathfrak{Q}$-машины) будет доказано в следующей главе. Там будет введена другая модель – LTS (Labelled Transition System), для которой мы определим множество её $\mathfrak{R}$-трасс. Будет показано, что 1) множество $\mathfrak{R}$-трасс LTS является $\mathfrak{R}$-моделью (это можно рассматривать как генетическое определение $\mathfrak{R}$-модели), 2) для каждой $\mathfrak{R}$-модели существует LTS-модель с этим множеством $\mathfrak{R}$-трасс и 3) любая LTS-модель в алфавите L может функционировать как реализация, помещённая в чёрный ящик любой $\mathfrak{R}/\mathfrak{Q}$-машины в алфавите L. В результате мы докажем эквивалентность этих двух моделей в смысле эквивалентности интенсионального и генетического (через LTS) определений $\mathfrak{R}$-модели и их связь с машиной тестирования (Рис.25).

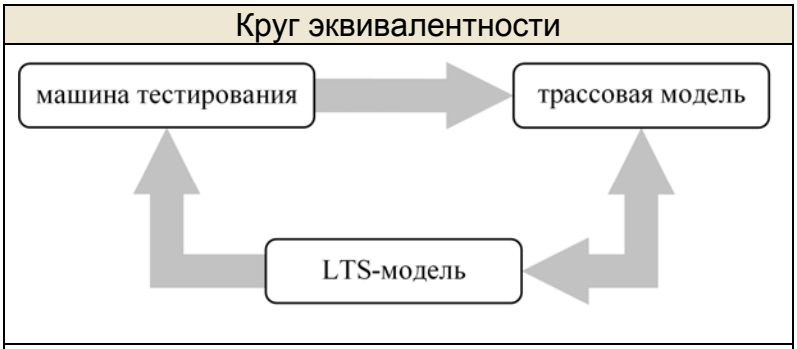


Рис.25.

Одной $\mathfrak{R}$-модели соответствует несколько LTS с тем же множеством $\mathfrak{R}$-трасс. В следующей главе мы рассмотрим два преобразования $\mathfrak{R}$-модели в LTS. Первое преобразование более простое, но строит, как правило, бесконечную LTS. Второе преобразование сложнее, но строит более «компактную» LTS, в частности, конечную, если такая существует. Для этого второго преобразования

нам понадобится утверждение о разложении $\mathfrak{R}$-модели на поддеревья, которое мы докажем в этой главе.

Далее мы рассмотрим безопасное тестирование. Будут формально определены отношения ***safe in*** и ***safe by***, гипотеза о безопасности (отношение ***safe for***) и безопасная конформность ***saco***.

После этого мы определим для трассовой модели понятия теста, набора тестов и отношение «реализация проходит тест», а также рассмотрим генерацию полного тестового набора.

3.1. Определение модели

Структура раздела:

1. Внешнее действие, разрушение
2. Отказы, $\mathfrak{R}/\mathfrak{Q}$-семантика и $\mathfrak{R}$-трассы
3. Свойства трасс и деревьев трасс
4. Определение $\mathfrak{R}$-модели

3.1.1. Внешнее действие, разрушение и дивергенция

Определение 6:

- Будем считать, что фиксировано множество – *универсум внешних действий* Z.
- *Алфавитом внешних действий* будем называть множество внешних действий $L \subseteq Z$. Если не оговорено противное, будем предполагать, что алфавит внешних действий фиксирован и обозначается L.
- *Разрушение* будем обозначать символом γ и называть также γ-действием. Предполагается, что $\gamma \notin Z$.
- *Дивергенцию* будем обозначать символом Δ и называть также Δ-действием. Предполагается, что $\Delta \notin Z$ и $\Delta \neq \gamma$.
- Для произвольного множества A обозначим:

$A_\gamma =_{def} A\cup\{\gamma\}$ и $A_\Delta =_{def} A\cup\{\Delta\}$.

- *Базовыми действиями* будем называть внешние действия из алфавита L, разрушение γ и дивергенцию Δ.
- В рамках данной теории внешние действия, разрушение и дивергенцию будем считать *праэлементами*, то есть объектами, не являющимися классами.

□

3.1.2. Отказы, $\mathfrak{R}/\mathfrak{Q}$-семантика и $\mathfrak{R}$-трассы

Определение 7:

- *Отказом* P будем называть подмножество алфавита внешних действий $P\in\mathcal{P}(L)$.
- Объединение семейства отказов $\mathfrak{R}\subseteq\mathcal{P}(L)$ и произвольного множества A будем обозначать $A_{\mathfrak{R}} =_{def} A\cup\mathfrak{R}$.
- Будем говорить, что в алфавите $L\subseteq Z$ задана $\mathfrak{R}/\mathfrak{Q}$-семантика, если заданы два непересекающихся семейства отказов $\mathfrak{R}\subseteq\mathcal{P}(L)$ и $\mathfrak{Q}\subseteq\mathcal{P}(L)$, $\mathfrak{R}\cap\mathfrak{Q}=\varnothing$, объединение которых покрывает алфавит внешних действий: $\cup(\mathfrak{R}\cup\mathfrak{Q})=\cup\mathfrak{R}\cup\cup\mathfrak{Q}=L$. Элементы семейства $\mathfrak{R}$ будем называть $\mathfrak{R}$-отказами или наблюдаемыми отказами, а элементы семейства $\mathfrak{Q}$ будем называть $\mathfrak{Q}$-отказами или ненаблюдаемыми отказами. Если не оговорено противное, будем предполагать, что $\mathfrak{R}/\mathfrak{Q}$-семантика фиксирована, и семейства $\mathfrak{R}$- и $\mathfrak{Q}$-отказов обозначаются $\mathfrak{R}$ и $\mathfrak{Q}$.

□

Определение 8: *$\mathfrak{R}$-трассой* будем называть последовательность базовых действий и $\mathfrak{R}$-отказов, то есть последовательность в алфавите $L_{\mathfrak{R}\Delta\gamma}=L\cup\mathfrak{R}\cup\{\Delta,\gamma\}$. Также $\mathfrak{R}$-трассу будем называть также *трассой наблюдений* или просто *трассой*.

□

Поскольку внешние действия, разрушение и дивергенция считаются праэлементами, в трассах они не путаются с отказами как множествами внешних действий.

Трассы будем обозначать строчными греческими буквами λ,μ,σ..., а их множества прописными греческими буквами $\Lambda,\mathbf{M},\Sigma,\ldots$,

Определение 9: Максимальные префикс и постфикс трассы, состоящие только из отказов, будем обозначать:

$$\textbf{\textit{pref}}(\sigma) =_{def} \langle\ \sigma(i)\ \mid\ \sigma[1..i]\in\mathcal{P}(L)^*\ \rangle \text{ и}$$

$$\textbf{\textit{postf}}(\sigma) =_{def} \langle\ \sigma(i)\ \mid\ \sigma[i..|\sigma|]\in\mathcal{P}(L)^*\ \rangle.$$

□

3.1.3. Свойства трасс и деревьев трасс

Определение 10: Трассу σ будем называть *допустимой*, если дивергенция и разрушение либо не входят в трассу, либо являются последним символом трассы:

$$\forall i\in[1..|\sigma|-1]\ \ \sigma(i)\notin\{\Delta,\gamma\}.$$

□

Определение 11: Трассу σ будем называть *согласованной*, если в ней любая непустая последовательность отказов не продолжается ни дивергенцией, ни разрушением, ни каким-либо внешним действием, принадлежащим какому-либо отказу, входящему в эту последовательность:

$\forall i \in [2..|\sigma|]\ \forall P \in \mathit{Im}_{\circ}\mathit{postf}(\sigma[1..i-1])\ \sigma(i) \notin P_{\Delta\gamma}$.

□

Определение 12: Пусть задано дерево трасс Σ и трасса $\sigma \in \Sigma$.

- Кнопку "P", где $P \in \mathfrak{R} \cup \mathfrak{Q}$, будем называть *разрушающей* после трассы σ, если $\exists z \in P\ \sigma \cdot \langle z, \gamma \rangle \in \Sigma$.
- Трассу σ будем называть *конвергентной по отказу* P или P-*конвергентной* (в дереве Σ), и обозначать $\sigma\downarrow P$, если трасса σ продолжается в дереве Σ этим отказом P или каким-либо внешним действием, принадлежащим отказу P. Если трасса не P-конвергентна, будем называть её P-*дивергентной* (в дереве Σ) и обозначать $\sigma\uparrow P$.

  $P \in \mathfrak{R}\ :\ \sigma\downarrow P\ =_{\text{def}}\ \sigma \cdot \langle P \rangle \in \Sigma\ \vee\ \exists z \in P\ \sigma \cdot \langle z \rangle \in \Sigma$,

  $P \in \mathfrak{R}\ :\ \sigma\uparrow P\ =_{\text{def}}\ \neg\sigma\downarrow P$.

- Трассу σ будем называть *($\mathfrak{R}$-)конвергентной* (в дереве Σ) и обозначать $\sigma\downarrow\mathfrak{R}$, если она конвергентна по всем $\mathfrak{R}$-отказам. В противном случае трассу будем называть *($\mathfrak{R}$-)дивергентной* (в дереве Σ), и обозначать $\sigma\uparrow\mathfrak{R}$.

  $\sigma\downarrow\mathfrak{R}\ =_{\text{def}}\ \forall P \in \mathfrak{R}\ \sigma\downarrow P$,

  $\sigma\uparrow\mathfrak{R}\ =_{\text{def}}\ \neg\sigma\downarrow\mathfrak{R}$.

□

Определение 13: Операции над трассами и замыкание по операциям.

- Определим операции над трассами:
  - ***d*** (***d***eletion) – удаление отказа:

    $P\in\mathfrak{R}$: $(\mu\cdot\langle P\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{d}\rightarrow\mu\cdot\lambda$.

  - ***r*** (***r***epetition) – повторение отказа:

    $P\in\mathfrak{R}$: $(\mu\cdot\langle P\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{r}\rightarrow\mu\cdot\langle P,P\rangle\cdot\lambda$.

  - ***t*** (***t***ransposition) – перестановка местами соседних отказов:

    $P,Q\in\mathfrak{R}$: $(\mu\cdot\langle P,Q\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{t}\rightarrow\mu\cdot\langle Q,P\rangle\cdot\lambda$.

- Под замыканием по набору этих операций будем понимать отображение трассы σ во множество трасс, получаемых из σ с помощью всех возможных последовательностей (в том числе пустой) применения операций из этого набора. Обозначим замыкание трассы σ по набору операций $\{\boldsymbol{op}_1,\ldots,\boldsymbol{op}_k\}\subseteq\{\boldsymbol{d},\boldsymbol{r},\boldsymbol{t}\}$:

  $\boldsymbol{op}_1\ldots\boldsymbol{op}_k(\sigma) =_{def} \{\sigma` \mid \exists n\in\mathcal{N}\ \exists\mu_1,\ldots,\mu_n\ \exists\sigma_1,\ldots,\sigma_n\ \sigma=\sigma_1\ \&\ \sigma`=\sigma_n$

  $\&\ \forall i\in[1..n-1]\ \exists j\in[1..k]\ (\sigma_i,\mu_i)\text{—}\boldsymbol{op}_j\rightarrow\sigma_{i+1}\}$.

□

Для любой трассы σ замыкание по любому набору операций из $\{\boldsymbol{d},\boldsymbol{r},\boldsymbol{t}\}$ можно делать последовательно: сначала по одной операции, потом по другой и т.д. Например, $\boldsymbol{dt}(\sigma)=\cup_\circ\boldsymbol{t}_\circ\boldsymbol{d}(\sigma)=\cup_\circ\boldsymbol{d}_\circ\boldsymbol{t}(\sigma)$. Соответственно, для множества трасс $\mathbf{T}$ всегда имеет место $\cup_\circ\boldsymbol{dt}(\mathbf{T})=\cup_\circ\boldsymbol{t}_\circ\cup_\circ\boldsymbol{d}(\mathbf{T})=\cup_\circ\boldsymbol{d}_\circ\cup_\circ\boldsymbol{t}(\mathbf{T})$.

Определение 14: Пусть задано дерево трасс $\boldsymbol{\Sigma}$.

- Определим операцию ***i*** (***i***nsertion) вставки отказа Q в трассу $\mu\cdot\lambda$ после трассы μ при условии, что трасса μ заканчивается отказом и не продолжается в $\boldsymbol{\Sigma}$ дивергенцией, разрушением и действиями из Q:

  $\boldsymbol{postf}(\mu)\neq\epsilon\ \&\ \mu\cdot\lambda\in\boldsymbol{\Sigma}\ \&\ Q\in\mathfrak{R}\ \&\ \forall u\in Q_{\Delta\gamma}\ \mu\cdot\langle u\rangle\notin\boldsymbol{\Sigma}$

  : $(\mu\cdot\lambda,\mu,Q)\text{—}\boldsymbol{i}\rightarrow\mu\cdot\langle Q\rangle\cdot\lambda$.

· Под замыканием трассы σ по операции $\boldsymbol{i}$ будем понимать отображение трассы σ во множество трасс, получаемых из σ с помощью всех возможных одновременных вставок любого (в том числе нулевого) числа отказов. Обозначим для трассы σ:

$\boldsymbol{i}(\sigma) =_{\text{def}} \{\ \sigma` \mid \exists n\in\mathcal{N}_0\ \exists \mu_0,\dots,\mu_n\ \exists Q_1,\dots,Q_n$

$\sigma=\mu_0\cdot\ldots\cdot\mu_n\ \&\ \sigma`=\mu_0\cdot\langle Q_1\rangle\cdot\mu_1\cdot\ldots\cdot\langle Q_n\rangle\cdot\mu_n\ \&$

$\forall i\in[1..n]\ (\sigma,\mu_0\cdot\ldots\cdot\mu_{i-1},Q_i)\text{—}\boldsymbol{i}\rightarrow(\mu_0\cdot\ldots\cdot\mu_{i-1})\cdot\langle Q_i\rangle\cdot(\mu_i\cdot\ldots\cdot\mu_n)\ \}.$

· Под замыканием по набору операций $\{\boldsymbol{op}_1,\dots,\boldsymbol{op}_k\}\cup\{\boldsymbol{i}\}$, где $\{\boldsymbol{op}_1,\dots,\boldsymbol{op}_k\}\subseteq\{\boldsymbol{d},\boldsymbol{r},\boldsymbol{t}\}$, будем понимать композицию замыкания по поднабору операций $\{\boldsymbol{op}_1,\dots,\boldsymbol{op}_k\}$ и замыкания по операции $\boldsymbol{i}$:

$\boldsymbol{op}_1\dots\boldsymbol{op}_k\boldsymbol{i}(\sigma) =_{\text{def}} \cup\circ\boldsymbol{op}_1\dots\boldsymbol{op}_k\circ\boldsymbol{i}(\sigma)$.

□

Все операции, кроме операции вставки, не зависят от семейства $\mathfrak{R}$ (вставляемых отказов) и дерева $\boldsymbol{\Sigma}$. Если замыкание строится по набору операций, включающих операцию $\boldsymbol{i}$, то там, где по контексту неясно, какое семейство $\mathfrak{R}$ и/или какое дерево имеется в виду, будем указывать их в нижнем индексе: например, $\boldsymbol{i}_{\mathfrak{R}}(\sigma)$ и $\boldsymbol{drti}_{\mathfrak{R}}(\sigma)$, $\boldsymbol{i}_{\text{in}\,\boldsymbol{\Sigma}}(\sigma)$ и $\boldsymbol{drti}_{\text{in}\,\boldsymbol{\Sigma}}(\sigma)$ или $\boldsymbol{i}_{\mathfrak{R}\ \text{in}\,\boldsymbol{\Sigma}}(\sigma)$ и $\boldsymbol{drti}_{\mathfrak{R}\ \text{in}\,\boldsymbol{\Sigma}}(\sigma)$.

3.1.4. Определение $\mathfrak{R}$-модели

Определение 15: Непустое дерево трасс $\boldsymbol{\Sigma}$ будем называть *моделью наблюдаемых трасс* или *$\mathfrak{R}$-моделью*, если выполнены следующие пять требований:

($\mathfrak{R}_1$) допустимость: все трассы $\boldsymbol{\Sigma}$ допустимы;

($\mathfrak{R}_2$) согласованность: все трассы $\boldsymbol{\Sigma}$ согласованы;

($\mathfrak{R}_3$) $\mathfrak{R}$-конвергентность: все трассы $\boldsymbol{\Sigma}$, не содержащие и не продолжающиеся разрушением и дивергенцией, $\mathfrak{R}$-конвергентны;

($\mathfrak{R}_4$) замкнутость: $\boldsymbol{\Sigma}$ замкнуто по $\boldsymbol{d}$-операции: $\cup\circ\boldsymbol{d}(\boldsymbol{\Sigma}) = \boldsymbol{\Sigma}$;

($\mathfrak{R}_5$) $\mathfrak{R}$-полнота: $\boldsymbol{\Sigma}$ замкнуто по $\boldsymbol{i}$-операции: $\cup\circ\boldsymbol{i}(\boldsymbol{\Sigma}) = \boldsymbol{\Sigma}$.

Множество всех $\mathfrak{R}$-моделей в алфавите $\mathtt{L}$ обозначим $\boldsymbol{MODEL}(\mathtt{L}_{\mathfrak{R}\Delta\gamma})$.

□

Свойства $\mathfrak{R}$-модели можно наглядно представить себе как свойства некоторого порождающего её графа[8]. Поскольку порождается дерево трасс, в графе все вершины конечные и одна начальная вершина (Теорема 1:). Свойство допустимости означает, что дуги, помеченные символами γ или Δ, ведут в терминальные вершины (вершины, из которых не выходят дуги). Замкнутость означает, что дуги, помеченные отказами, – это петли. Стабильной вершиной будем называть вершину, из которой выходят только дуги, помеченные внешними действиями, то есть не выходят непомеченные дуги и дуги, помеченные символами γ или Δ. Согласованность означает, что петли, помеченные отказами, определяются только в стабильных вершинах. Кроме того, из таких вершин не могут выходить дуги, помеченные действиями, принадлежащими каким-либо отказам, которыми помечены петли в этих вершинах. $\mathfrak{R}$-полнота означает, что, если из стабильной вершины не выходят дуги, помеченные действиями из некоторого $\mathfrak{R}$-отказа, то в этой вершине определена петля, помеченная этим $\mathfrak{R}$-отказом. $\mathfrak{R}$-конвергентность означает, что в графе нет бесконечного маршрута (в частности, маршрута, порождаемого циклом), содержащего только непомеченные дуги.

---

[8] Здесь мы имеем в виду только конечные последовательности, порождаемые графом.

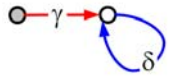
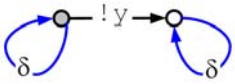
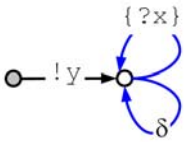
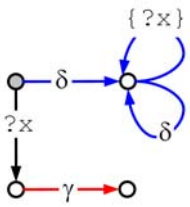
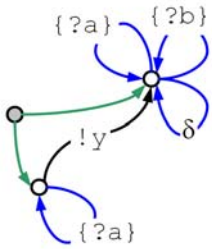
Можно доказать, что множество трасс наблюдений является $\mathfrak{R}$-моделью тогда и только тогда, когда существует порождающий это множество граф с указанными свойствами. Мы не будем давать формального доказательства этого утверждения. Вместо этого мы в следующей главе докажем два утверждения о связи $\mathfrak{R}$-модели с LTS: 1) множество $\mathfrak{R}$-трасс LTS является $\mathfrak{R}$-моделью и 2) каждая $\mathfrak{R}$-модель является множеством $\mathfrak{R}$-трасс некоторой LTS. Последняя LTS отличается от порождающего графа с указанными свойствами только следующим: вершины называются состояниями; дуги называются переходами; вместо непомеченных дуг используются переходы, помеченные символом τ; петли отказов не определяются, поскольку в нестабильных состояниях их нет, а в стабильных состояниях они однозначно восстанавливаются по множеству внешних действий, которыми помечены выходящие из состояния переходы; вместо Δ-дуги определяется τ-петля (дивергенция).

Заметим также, что не любое дерево допустимых $\mathfrak{R}$-трасс является $\mathfrak{R}$-моделью. Исключение составляет только вырожденный случай, когда нет $\mathfrak{R}$-кнопок (и, следовательно, $\mathfrak{R}$-отказов): любое дерево допустимых $\varnothing$-трасс является трассовой моделью. Допустимость предполагается, а выполнение остальных четырёх свойств очевидно.

Не считая того, что в $\mathfrak{R}$-трассах все отказы из семейства $\mathfrak{R}$, среди пяти свойств $\mathfrak{R}$-модели только два непосредственно зависят от семейства $\mathfrak{R}$: конвергентность и полнота.

Пять свойств $\mathfrak{R}$-модели независимы в следующем смысле: существует алфавит внешних действий $L \subseteq Z$ и семейство $\mathfrak{R} \subseteq \mathcal{P}(L)$ такие, что для некоторых деревьев $\mathfrak{R}$-трасс могут выполняться все свойства, кроме одного. Примеры показаны на Рис.26 для $\beta\gamma\delta$-семантики: L состоит из стимулов и

реакций, а $\mathfrak{R}=\{\delta\}\cup\{\{?x\}\mid ?x\in L\}$, где $\delta=\{!y\mid !y\in L\}$. Её $\mathfrak{R}$-трассы будем называть βγδ-трассами, а её $\mathfrak{R}$-модели – βγδ-моделями, вместо $\mathfrak{R}$ будем писать βδ. Здесь для каждого i-го свойства ($i=1\div5$) изображён порождающий граф дерева βγδ-трасс, удовлетворяющего всем свойствам βγδ-модели, кроме i-го.

<table>
<tr><th colspan="5">Независимость свойств $\mathfrak{R}$-модели на примере βγδ-модели Σ</th></tr>
<tr><td>1.
допустимость</td><td>2.
согласованность</td><td>3.
конвергентность</td><td>4.
замкнутость</td><td>5.
полнота</td></tr>
<tr><td colspan="2">$\mathfrak{R}=\{\delta\}=\{\{!y\}\}$,
$\mathfrak{Q}=\varnothing$</td><td colspan="2">$\mathfrak{R}=\{\delta\}=\{\{!y\}\}$,
$\mathfrak{Q}=\{\{?x\}\}$</td><td>$\mathfrak{R}=\{\delta\}=\{\{!y\}\}$,
$\mathfrak{Q}=\{\{?a\},\{?b\}\}$</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>$\langle\gamma,\delta\rangle\in\Sigma$, но
не допустима</td><td>$\langle\delta,!y\rangle\in\Sigma$, но
не согласована</td><td>$\epsilon\in\Sigma$, но
$\epsilon\downarrow\{?x\}$</td><td>$\sigma=\langle\delta,\{?x\}\rangle\in\Sigma$,
$\langle\{?x\}\rangle\in\boldsymbol{d}(\sigma)\setminus\Sigma$</td><td>$\sigma=\langle\{?a\},!y\rangle\in\Sigma$,
$\langle\{?a\},\{?b\},!y\rangle\in\boldsymbol{i}(\sigma)\setminus\Sigma$</td></tr>
<tr><td colspan="5">Рис.26.</td></tr>
</table>

Из условия согласованности вытекает следующее свойство *ослабленной согласованности*: $\forall i\in[2..|\sigma|]\ \sigma(i-1)\in\mathcal{P}(L)\ \Rightarrow\ \sigma(i)\notin\sigma(i-1)_{\Delta\gamma}$.

При условии замкнутости это свойство эквивалентно согласованности. Действительно, если бы была трасса $\sigma\cdot\langle P\rangle\cdot\langle Q_1\rangle\cdot\ldots\cdot\langle Q_n\rangle\cdot\langle z\rangle$, где P и $Q_1,\ldots,Q_n$ отказы, а $z\in P_{\Delta\gamma}$, то, по замкнутости, была бы и трасса $\sigma\cdot\langle P\rangle\cdot\langle z\rangle$, а такая трасса запрещена условием ослабленной согласованности.

Из свойств согласованности и $\mathfrak{R}$-полноты непосредственно следует замкнутость модели по операции ***r*** повторения отказов и операции ***t***

перестановки отказов. Действительно, префикс $\mu\cdot\langle \mathtt{P}\rangle$ трассы $\mu\cdot\langle \mathtt{P}\rangle\cdot\lambda\in\boldsymbol{\Sigma}$, заканчивается отказом $\mathtt{P}$, и поэтому не может, по согласованности, продолжаться в $\mathfrak{R}$-модели дивергенцией, разрушением или внешним действием из $\mathtt{P}$ и, следовательно, по $\mathfrak{R}$-полноте, отказ $\mathtt{P}$ можно повторить в трассе: $\mu\cdot\langle \mathtt{P},\mathtt{P}\rangle\cdot\lambda\in\boldsymbol{\Sigma}$. Также префикс $\mu\cdot\langle \mathtt{P},\mathtt{Q}\rangle$ трассы $\mu\cdot\langle \mathtt{P},\mathtt{Q}\rangle\cdot\lambda\in\boldsymbol{\Sigma}$ не может, по согласованности, продолжаться в $\mathfrak{R}$-модели дивергенцией, разрушением или внешним действием из $\mathtt{P}$ и, следовательно, по $\mathfrak{R}$-полноте, отказ $\mathtt{P}$ можно вставить после этого префикса: $\mu\cdot\langle \mathtt{P},\mathtt{Q},\mathtt{P}\rangle\cdot\lambda\in\boldsymbol{\Sigma}$. А тогда, удаляя первый отказ $\mathtt{P}$, получаем трассу с переставленными отказами $\mu\cdot\langle \mathtt{Q},\mathtt{P}\rangle\cdot\lambda\in\boldsymbol{\Sigma}$. Таким образом, свойство замкнутости может быть заменено на следующее свойство *усиленной замкнутости*: $\boldsymbol{drt}(\sigma)\subseteq\boldsymbol{\Sigma}$.

Усиленная замкнутость и $\mathfrak{R}$-полнота означают замкнутость по ***drti***-операциям: $\forall\sigma\in\boldsymbol{\Sigma}\ \ \boldsymbol{drti}(\sigma)\subseteq\boldsymbol{\Sigma}$. Поскольку для любой трассы $\sigma\cdot\lambda\in\boldsymbol{\Sigma}$ и любой трассы $\sigma`\in\boldsymbol{drti}(\sigma)$ имеет место $\sigma`\cdot\lambda\in\boldsymbol{drti}(\sigma\cdot\lambda)$, из усиленной замкнутости и $\mathfrak{R}$-полноты выводится очевидное следствие: любое продолжение трассы продолжает каждую трассу, получаемую из данной трассы ***drti***-операциями: $\forall\sigma\in\boldsymbol{\Sigma}\ \ \boldsymbol{drti}(\sigma)\cdot(\boldsymbol{\Sigma}\ \ \boldsymbol{after}\ \ \sigma)\subseteq\boldsymbol{\Sigma}$.

3.2. Полная модель (*F*-модель)

Структура раздела:

1. *F*-модель и её $\mathfrak{R}$-проекция
2. Расширение $\mathfrak{R}$-модели до *F*-модели

Сейчас мы введём понятие ***F**-модели*, проекцией которой на выбранное семейство отказов $\mathfrak{R}$ оказывается $\mathfrak{R}$-модель. Можно считать, что ***F***-модель является «полной моделью» и описывает, как устроена система «на самом деле». Кавычки здесь нужны, поскольку речь идёт об описании на некотором уровне абстракции, когда поведение системы формализуется в виде последовательности дискретных внешних действий из фиксированного алфавита $\mathtt{L}$ или отсутствия таковых (отказ), а также разрушения и дивергенции.

$\mathfrak{R}$-модель можно трактовать как «взгляд» на систему, определяемый тестовыми возможностями по управлению и наблюдению, описываемыми $\mathfrak{R}/\mathfrak{Q}$-машиной. Такая ***F***-модель может помещаться в чёрный ящик любой $\mathfrak{R}/\mathfrak{Q}$-машины. Каждая $\mathfrak{R}/\mathfrak{Q}$-машина определяет $\mathfrak{R}$-проекцию ***F***-модели как множество наблюдаемых трасс.

С другой стороны, ***F***-модель всё же абстрактнее LTS-модели, которую мы рассмотрим в следующей главе, поскольку описание остаётся на уровне наблюдаемых трасс. Только такие наблюдения в случае ***F***-модели мы можем делать не на фиксированной $\mathfrak{R}/\mathfrak{Q}$-машине, а на всей их совокупности, или одной ***F***-машине, то есть $\mathcal{P}(\mathtt{L})/\varnothing$-машине. Если не считать введённых нами дивергенции и разрушения, семантика ***F***-модели совпадает с семантикой трасс с отказами (*failure trace semantics*), но задаётся нами с помощью

интенсионального, а не генетического (через LTS), определения, как это обычно делают [89,90,149].

3.2.1. *F*-модель и её $\mathfrak{R}$-проекция

Определение 16: ***F***-модель и её $\mathfrak{R}$-проекция.

- Для $\mathfrak{R}=\mathcal{P}(\mathrm{L})$ будем называть:
  - $\mathfrak{R}$-трассу – ***F**-трассой*,
  - $\mathfrak{R}$-модель – ***F**-моделью*,
  - $\mathfrak{R}$-отказ – ***F**-отказом* или просто *отказом*.
- ***$\mathfrak{R}$-проекцией*** ***F***-модели $\boldsymbol{\Sigma}$, где $\mathfrak{R}\subseteq\mathcal{P}(\mathrm{L})$ семейство отказов, будем называть подмножество её $\mathfrak{R}$-трасс, то есть трасс, содержащих отказы только из семейства $\mathfrak{R}$: $\boldsymbol{\Sigma}\cap\mathrm{L}_{\mathfrak{R}\Delta\gamma}{}^{*}$.
- Множество ***F***-моделей в алфавите L обозначим
  $\boldsymbol{FMODEL}(\mathrm{L}) =_{\mathrm{def}} \boldsymbol{MODEL}(\mathrm{L}_{\mathcal{P}(\mathrm{L})\Delta\gamma})$.

□

Заметим, что, в силу замкнутости ***F***-модели по операции удаления отказов, подмножество $\mathfrak{R}$-трасс ***F***-модели совпадает с множеством $\mathfrak{R}$-проекций её трасс: $\boldsymbol{\Sigma}\cap\mathrm{L}_{\mathfrak{R}\Delta\gamma}{}^{*} = \boldsymbol{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}\Delta\gamma}$.

Теорема 2: $\mathfrak{R}$-проекция ***F***-модели является $\mathfrak{R}$-моделью.

□436

3.2.2. Расширение $\mathfrak{R}$-модели до *F*-модели

Любая $\mathfrak{R}$-модель Σ может быть расширена до ***F***-модели $\Sigma`$ так, что $\Sigma`^{\downarrow \mathtt{L}_{\mathfrak{R}\Delta\gamma}}=\Sigma$. Идея этого расширения состоит в следующем. Если нет пустой $\mathfrak{R}$-кнопки, определим операцию $\boldsymbol{e}$ вставки пустого отказа в трассу $\mu\cdot\lambda$ после трассы μ при условии, что μ не заканчивается и не продолжается в Σ дивергенцией, разрушением и отказами. Замыкание по этой операции сохраняет допустимость и согласованность трасс, и каждая трасса такого замыкания, не заканчивающаяся и не продолжающаяся дивергенцией и разрушением, заканчивается или продолжается отказом. При наличии пустой $\mathfrak{R}$-кнопки под ***e***-замыканием будем понимать пустую операцию. Потом сделаем замыкание по операции $\boldsymbol{i}_{\mathcal{P}(\mathtt{L})}$ вставки недостающих отказов, после чего выполним замыкание по ***d***-операции.

Пример, иллюстрирующий связь $\mathfrak{R}$-модели и ***F***-модели приведён на Рис.27. Расширению модели соответствует добавление петель по отказам из $\mathcal{P}(\mathtt{L})\setminus\mathfrak{R}$ в стабильных вершинах порождающего графа.

F-модели и $\mathfrak{R}$-проекции	
$\mathtt{L}=\{a,b,c\}$, $\mathfrak{R}=\{\{a\},\{b\},\{b,c\}\}$	
Σ	$\Sigma_{\mathfrak{R}}$

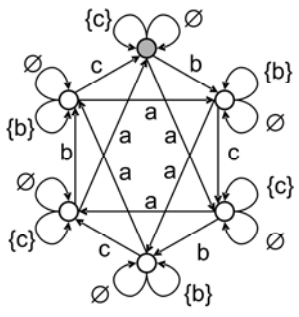
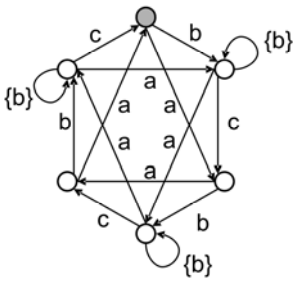

	

Рис.27.	

Определение 17: Пусть задано дерево $\mathfrak{R}$-трасс Σ.

· Пусть $\varnothing \notin \mathfrak{R}$.

· Определим операцию $\boldsymbol{e}$ вставки пустого отказа в трассу $\sigma=\mu\cdot\lambda$ после μ при условии, что трасса $\mu\cdot\lambda\in\Sigma$, а μ не заканчивается и не продолжается в Σ дивергенцией, разрушением и отказами:

$$\mu\cdot\lambda\in\Sigma \;\&\; (\mu\neq\epsilon \Rightarrow \mu(|\mu|)\in L) \;\&\; \forall u\notin L\ \ \mu\cdot\langle u\rangle\notin\Sigma$$

$$: \ (\mu\cdot\lambda,\mu)\text{—}\boldsymbol{e}\rightarrow\mu\cdot\langle\varnothing\rangle\cdot\lambda.$$

· Под замыканием трассы σ по операции $\boldsymbol{e}$ будем понимать отображение трассы σ во множество трасс, получаемых из σ с помощью всех возможных одновременных вставок любого (в том числе нулевого) числа пустых отказов. Обозначим для трассы σ:

$$\boldsymbol{e}(\sigma) =_{def} \{\sigma` \mid \exists n\in\mathcal{N}_0\ \exists\mu_0,\dots,\mu_n$$

$$\sigma=\mu_0\cdot\dots\cdot\mu_n \;\&\; \sigma`=\mu_0\cdot(\langle\varnothing\rangle\cdot\mu_1\cdot\dots\cdot\langle\varnothing\rangle\cdot\mu_n) \;\&$$

$$\forall i\in[1..n]\ \ (\sigma,\mu_0\cdot\dots\cdot\mu_{i-1})\text{—}\boldsymbol{e}\rightarrow(\mu_0\cdot\dots\cdot\mu_{i-1})\cdot\langle\varnothing\rangle\cdot(\mu_i\cdot\dots\cdot\mu_n)\}.$$

· Для случая $\varnothing\in\mathfrak{R}$ определим:

$$\boldsymbol{e}(\sigma) =_{def} \{\sigma\}.$$

· Расширением $\mathfrak{R}$-модели Σ до $\boldsymbol{F}$-модели назовём преобразование

$$\boldsymbol{Ext}(\Sigma) =_{def} \cup\circ\boldsymbol{d}\circ\cup\circ\boldsymbol{i}_{\mathcal{P}(L)}\circ\cup\circ\boldsymbol{e}(\Sigma).$$

□

Теорема 3: Для $\mathfrak{R}$-модели Σ её расширение $\boldsymbol{Ext}(\Sigma)$ является $\boldsymbol{F}$-моделью, и $\boldsymbol{Ext}(\Sigma)^{\downarrow L}{}_{\mathfrak{R}\Delta\gamma}=\Sigma$.

□437

3.3. Объединение и пересечение моделей

Структура раздела:

1. Объединение моделей
2. Пересечение моделей

3.3.1. Объединение моделей

Теорема 4: Объединение любого множества ***F***-моделей в одном алфавите является ***F***-моделью.

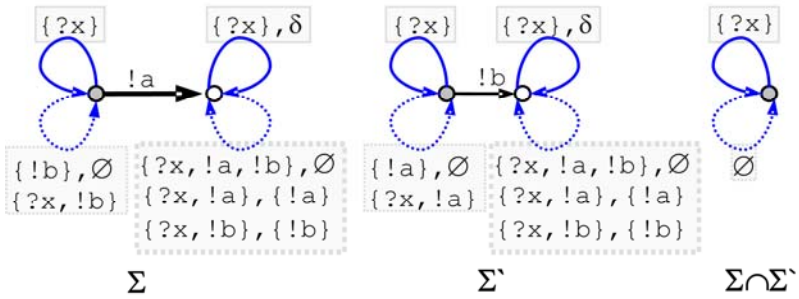
□439

F-модели в разных алфавитах L_i (где i пробегает некоторое множество индексов I) имеют разные семейства наблюдаемых отказов $\mathcal{P}(L_i)$. Для объединения таких ***F***-моделей каждую из них нужно сначала, понимая как $\mathcal{P}(L_i)$-модель в объединении алфавитов $L=\cup\{L_i \mid i\in I\}$, расширить до ***F***-модели в алфавите L. При фиксированном семействе $\mathfrak{R}$, очевидно, $\mathfrak{R}$-проекция объединения множеств трасс равна объединению $\mathfrak{R}$-проекций этих множеств. Поскольку $\mathfrak{R}$-модель является $\mathfrak{R}$-проекцией некоторой ***F***-модели, объединение любого множества $\mathfrak{R}$-моделей также является $\mathfrak{R}$-моделью, совпадающей с $\mathfrak{R}$-проекцией объединения соответствующих ***F***-моделей. Здесь уже не важно, в одном или в разных алфавитах заданы эти $\mathfrak{R}$-модели, поскольку $\mathfrak{R}$-модель не меняется при расширении алфавита. Здесь важно лишь, чтобы объединяемые модели имели одно и то же семейство $\mathfrak{R}$.

3.3.2. Пересечение моделей

Пересечение ***F***-моделей в одном алфавите или $\mathfrak{R}$-моделей для одного семейства $\mathfrak{R}$, вообще говоря, не является ***F***-моделью или $\mathfrak{R}$-моделью, соответственно. Свойства допустимости, согласованности и замкнутости сохраняются в пересечении, но свойства $\mathcal{P}(L)$- или $\mathfrak{R}$-конвергентности и $\mathcal{P}(L)$- или $\mathfrak{R}$-полноты могут нарушиться. Пример в виде порождающих графов приведён на Рис.28 для βγδ-моделей. Здесь трасса $\langle\{?x\}\rangle$ заканчивается отказом, но в пересечении не продолжается ни реакцией, ни стационарностью. Это остаётся верным и для расширений до ***F***-моделей (показано пунктиром).

Пересечение моделей может не быть моделью



$\mathfrak{R}=\{\delta\}=\{\{!a,!b\}\}$,

$\mathfrak{Q}=\{\{?x\}\}$

Σ, **Σ`** βγδ-модели

Σ∩Σ` не βγδ-модель

Рис.28.

3.4. Машина тестирования и трассовая модель

Теорема 5: Множество наблюдаемых трасс $\mathfrak{R}/\mathfrak{Q}$-машины является $\mathfrak{R}$-моделью (Рис.29).

□440

Множество наблюдаемых трасс машины тестирования является $\mathfrak{R}$-моделью

Рис.29.

Из этого утверждения и Теорема 3: следует, что для любой $\mathfrak{R}/\mathfrak{Q}$-машины существует такая ***F***-модель, $\mathfrak{R}$-проекция которой совпадает с множеством наблюдаемых трасс этой машины.

В следующей главе мы докажем обратное утверждение: любая ***F***-модель может «функционировать» как тестируемая система в любой $\mathfrak{R}/\mathfrak{Q}$-машине. При этом множество наблюдаемых трасс такой $\mathfrak{R}/\mathfrak{Q}$-машины окажется совпадающим с $\mathfrak{R}$-проекцией ***F***-модели, а ненаблюдаемый $\mathfrak{Q}$-отказ будет возникать после $\mathfrak{R}$-трассы тогда и только тогда, когда эта $\mathfrak{R}$-трасса продолжается этим $\mathfrak{Q}$-отказом в ***F***-модели.

3.5. Разложение трассовой модели на поддеревья

Структура раздела:

1. Стабильные и контрстабильные деревья и квази-модели
2. Вспомогательные утверждения
3. Утверждение о разложении трассовой модели

В этом разделе мы покажем, что $\mathfrak{R}$-модель раскладывается в объединение контрстабильного дерева и множества стабильных деревьев (Рис.30).

Разложение модели на деревья

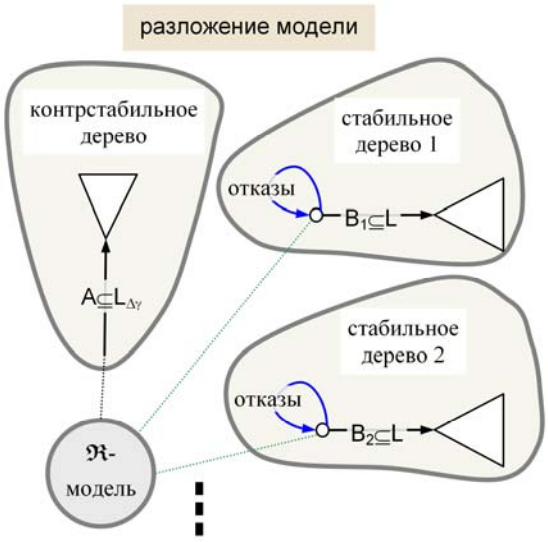


Рис.30.

Саму ℜ-модель можно рассматривать как модель поведения ℜ/𝔔-машины. Контрстабильное дерево описывает внешние действия, возможные в машине, вместе со всеми продолжающими эти действия поведениями машины, а также возможные дивергенцию и разрушение. Стабильное дерево является аналогом стабильного состояния ℜ/𝔔-машины, то есть состояния, когда машина стоит, вместе со всеми возможными дальнейшими поведениями машины. Подмодель, определяющая поведение машины после внешнего действия, также раскладывается в свою очередь, на контрстабильное дерево и множество стабильных поддеревьев. И так далее.

Поскольку ℜ-модель является ℜ-проекцией ***F***-модели, нам достаточно определить разложение ***F***-модели. Тогда разложение ℜ-модели является ℜ-проекцией разложения ***F***-модели, ℜ-проекцией которой является данная ℜ-модель.

Целью разложения ℜ-модели является построение LTS-модели с тем же множеством ℜ-трасс. Это будет сделано в следующей главе с помощью специального преобразования, создающего самую «компактную» LTS-модель:

***F2**$_{compact}$**L** : **F**-модель→LTS-модель. В частности, если для данной **F**- или $\mathfrak{R}$-модели существует конечная LTS с таким же множеством **F**- или $\mathfrak{R}$-трасс, то таким преобразованием будет получена конечная LTS.

3.5.1. Стабильные и контрстабильные деревья и квази-модели

Определение 18: Пусть задано дерево ***F***-трасс **T**.

· Дерево **T** 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**:

$\forall \rho \in \mathcal{P}(L)^* \cap \mathbf{T}$ **T** ***after*** ρ = **T**;

(стаб.2) в дереве **T** нет Δ- и γ-трасс, а его пустая трасса $\mathcal{P}(L)$-конвергентна: для каждого отказа есть трасса, начинающаяся с этого отказа или внешнего действия из этого отказа: $\langle\Delta\rangle \notin \mathbf{T}$ & $\langle\gamma\rangle \notin \mathbf{T}$ & $\epsilon \downarrow \mathcal{P}(L)$.

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

$\forall P \in \mathcal{P}(L) \cap$ ***head***(**T**) $\forall z \in P$ $z \notin$ ***head***(**T**).

· Если дерево **T** не стабильно, будем обозначать

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

□

Определение 19: Пусть задано дерево ***F***-трасс **T**.

· Дерево **T**, в котором ни одна трасса не начинается с отказа, будем называть *контрстабильным*:

head(**T**) $\subseteq L_{\Delta\gamma}$.

· Для дерева $\mathbf{T}$ через $\mathbf{T}_c$ обозначим его наибольшее (по вложенности) контрстабильное поддерево:

$\mathbf{T}_c =_{def} \{\lambda \in \mathbf{T} \mid \textit{pref}(\lambda) = \epsilon\}$.

□

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 удовлетворяет условию стаб.2, и не любое поддерево стабильного дерева стабильно.

Определение 20: Пусть задано дерево ***F***-трасс $\mathbf{T}$.

· Квази-моделью будем называть такое контрстабильное дерево $\mathbf{T}$, которое после каждого внешнего действия является ***F***-моделью, а дивергенция и разрушение ничем не продолжаются:

$\forall z \in \textit{head}(\mathbf{T}) \cap L \ (\mathbf{T} \ \textit{after} \ \langle z \rangle) \in \textit{FMODEL}(L)$

$\& \ \mathbf{T} \ \textit{after} \ \langle \Delta \rangle = \mathbf{T} \ \textit{after} \ \langle \gamma \rangle = \{\epsilon\}$.

□

3.5.2. Вспомогательные утверждения

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

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 (Рис.31).

Лемма 1: Пусть задано дерево ***F***-трасс $\mathbf{T}$, для которого выполнено свойство (стаб.1). Тогда $\mathbf{T}=O^{*}\cdot\mathbf{T}_{c}$, где $O=\boldsymbol{head}(\mathbf{T})\cap\mathcal{P}(L)$ – множество отказов, с которых могут начинаться трассы дерева $\mathbf{T}$, а $\mathbf{T}_{c}$ – наибольшее контрстабильное поддерево $\mathbf{T}$.

□442

Дерево после отказов	Модель после трассы

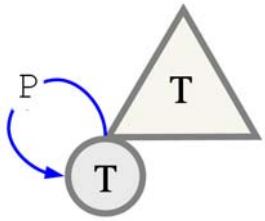
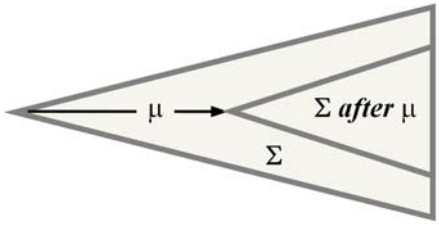

	

Рис.31.	Рис.32.

Покажем, что после каждой трассы, не заканчивающейся на дивергенцию или разрушение, ***F***-модель остаётся ***F***-моделью (Рис.32).

Лемма 2: Пусть задана ***F***-модель $\mathbf{\Sigma}$. Тогда для каждой трассы $\mu\in\mathbf{\Sigma}$, не заканчивающейся на дивергенцию или разрушение, имеем $(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu)\in\boldsymbol{FMODEL}(L)$.

□443

Лемма 3: Наибольшее контрстабильное поддерево ***F***-модели является квази-моделью.

□444

Квази-модель, не содержащую трасс $\langle\Delta\rangle$ и $\langle\gamma\rangle$, всегда можно расширить до ***F***-модели, сделав пустую трассу $\mathcal{P}(L)$-конвергентной: добавить отказы, если трассы не начинаются с внешних действий из этих отказов (в том числе всегда добавляется пустой отказ).

Лемма 4: Пусть задана квази-модель $\mathbf{T}$, не содержащая трасс $\langle\Delta\rangle\notin\mathbf{T}$ и $\langle\gamma\rangle\notin\mathbf{T}$. Тогда $O^*\cdot\mathbf{T}\in \boldsymbol{FMODEL}(L)$, где $O=\mathcal{P}(L\setminus \boldsymbol{head}(\mathbf{T}))$.

□445

3.5.3. Утверждение о разложении трассовой модели

Теорема 6: Пусть задана $\boldsymbol{F}$-модель $\boldsymbol{\Sigma}$. Тогда $\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_C\cup\cup\boldsymbol{\Sigma}_S$, где $\boldsymbol{\Sigma}_C$ (наибольшее по вложенности контрстабильное поддерево $\boldsymbol{\Sigma}$) является квази-моделью, а $\boldsymbol{\Sigma}_S$ – множество стабильных $\boldsymbol{F}$-моделей.

□446

$\mathfrak{R}$-проекция объединения множеств трасс равна объединению $\mathfrak{R}$-проекций этих множеств. $\mathfrak{R}$-проекция $\boldsymbol{F}$-модели является $\mathfrak{R}$-моделью, и любая $\mathfrak{R}$-модель является $\mathfrak{R}$-проекцией некоторой $\boldsymbol{F}$-модели. Поэтому утверждение о разложении трассовой модели верно и для $\mathfrak{R}$-модели.

Для дальнейшего нам также понадобится следующее простое утверждение: замыкание непустой трассы отказов ρ по ***drti***-операциям равно множеству всех конечных последовательностей в алфавите, содержащем все те отказы, для которых трасса ρ не продолжается в $\boldsymbol{F}$-модели ни одним внешним действием из отказа.

Лемма 5: Пусть задана $\boldsymbol{F}$-модель $\boldsymbol{\Sigma}$ и непустая трасса отказов $\rho\in\mathcal{P}(L)^*\cap\boldsymbol{\Sigma}$, $\rho\neq\epsilon$. Тогда $\boldsymbol{drti}(\rho) = \mathcal{P}(L\setminus \boldsymbol{head}(\boldsymbol{\Sigma}\ \boldsymbol{after}\ \rho))^*$.

□451

3.6. Безопасность и конформность

Структура раздела:

1. Безопасность в реализации
2. Безопасность в спецификации
3. Гипотеза о безопасности
4. Безопасная конформность

В этом разделе мы формально определим безопасность и конформность, не повторяя аргументацию предыдущей главы. Для заданной $\mathfrak{R}/\mathfrak{Q}$-семантики и ***F***-моделей реализации $\mathbf{I}$ и спецификации $\mathbf{\Sigma}$ мы должны определить:

- отношение `P` ***safe in*** $\mathbf{I}$ ***after*** σ, означающее «кнопка `P` безопасна в реализации $\mathbf{I}$ после трассы σ»;
- три правила, которым должно удовлетворять отношение `P` ***safe by*** $\mathbf{\Sigma}$ ***after*** σ, задающее протокол взаимодействия и означающее «кнопка `P` безопасна в спецификации $\mathbf{\Sigma}$ после трассы σ»;
- множество ***SafeIn***$(\mathbf{I})$ безопасных трасс реализации $\mathbf{I}$;
- множество ***SafeBy***$(\mathbf{\Sigma})$ безопасных трасс спецификации $\mathbf{\Sigma}$;
- отношение $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$, соответствующее гипотезе о безопасности и означающее, что реализация $\mathbf{I}$ безопасна для спецификации $\mathbf{\Sigma}$ при взаимодействии по протоколу, задаваемому отношением ***safe by***;
- конформность $\mathbf{I}$ ***saco*** $\mathbf{\Sigma}$ для протокола взаимодействия, задаваемого отношением ***safe by***.

Под безопасным тестированием здесь, как и в предыдущей главе, понимается тестирование, которое не может вызвать разрушение реализации, ненаблюдаемый отказ или отсутствие наблюдения из-за дивергенции.

Можно заметить, что, хотя все определения делаются для ***F***-моделей, фактически, используются только $\mathfrak{R}\cup\mathfrak{Q}$-проекция реализации $\mathbf{I}^{\downarrow}\mathrm{L}_{\mathfrak{R}\cup\mathfrak{Q}\gamma}$ и $\mathfrak{R}$-проекция спецификации $\mathbf{\Sigma}^{\downarrow}\mathrm{L}_{\mathfrak{R}\Delta\gamma}$.

3.6.1. Безопасность в реализации

Определение 21: Пусть задана ***F***-модель $\mathbf{I}$, которую мы будем называть *(**F**-)реализацией*.

- Кнопка "$\mathtt{P}$" *безопасна в реализации* $\mathbf{I}$ после $\mathfrak{R}$-трассы $\sigma\in\mathbf{I}^{\downarrow}\mathrm{L}_{\mathfrak{R}}$, если она не разрушающая, не вызывает ненаблюдаемого $\mathfrak{Q}$-отказа и после трассы нет дивергенции:

  $\mathtt{P}$ ***safe in*** $\mathbf{I}$ ***after*** σ

  $=_{\text{def}}$ $(\mathtt{P}\in\mathfrak{R} \vee \sigma\cdot\langle\mathtt{P}\rangle\notin\mathbf{I})$ & $\forall\mathtt{z}\in\mathtt{P}\ \sigma\cdot\langle\mathtt{z},\gamma\rangle\notin\mathbf{I}$ & $\sigma\cdot\langle\Delta\rangle\notin\mathbf{I}$.

- Внешнее действие $\mathtt{z}\in\mathrm{L}$ *безопасно в реализации* $\mathbf{I}$ после $\mathfrak{R}$-трассы $\sigma\in\mathbf{I}$, если оно разрешается какой-нибудь кнопкой, безопасной после σ:[9]

  $\mathtt{z}$ ***safe in*** $\mathbf{I}$ ***after*** σ $=_{\text{def}}$ $\exists\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}$ $\mathtt{z}\in\mathtt{P}$ & $\mathtt{P}$ ***safe in*** $\mathbf{I}$ ***after*** σ.

- $\mathfrak{R}$-трассу реализации $\mathbf{I}$ будем называть безопасной, если реализация не содержит трассу $\langle\gamma\rangle$, а трасса не заканчивается на дивергенцию и разрушение, и каждый встречающийся в ней символ $\mathtt{u}$ (внешнее действие или $\mathfrak{R}$-отказ) безопасен после непосредственно предшествующего ему префикса трассы:

---

[9] Заметим, что мы не требуем, чтобы трасса обязательно продолжалась в реализации безопасным внешним действием $\mathtt{z}$.

SafeIn $(\mathbf{I}) =_{\text{def}} \{\sigma \in \mathbf{I}^{\downarrow L_{\mathfrak{R}}} \mid \langle\gamma\rangle \notin \mathbf{I} \ \&\ \forall \mu\ \forall u$

$(\mu \cdot \langle u \rangle \le \sigma \Rightarrow u$ ***safe in*** $\mathbf{I}$ ***after*** $\mu)\}$.

□

Заметим, что, хотя отношение ***safe in*** формально определено на всех $\mathfrak{R}$-трассах реализации, не заканчивающихся дивергенцией и разрушением, для дальнейшего достаточно считать его определённым только на безопасных трассах. Иными словами, если трасса μ опасна, то для безопасной реализации нас не будет интересовать, какие кнопки безопасны или опасны после любого продолжения $\mu \cdot \lambda$.

3.6.2. Безопасность в спецификации

Определение 22: Пусть задана ***F***-модель $\mathbf{\Sigma}$.

- Будем говорить, что задан *протокол взаимодействия*, если задано отношение P ***safe by*** $\mathbf{\Sigma}$ ***after*** σ, означающее «кнопка "P" безопасна в спецификации $\mathbf{\Sigma}$ после $\mathfrak{R}$-трассы σ», удовлетворяющее трём *правилам протокола*: $\forall \sigma \in \mathbf{\Sigma}^{\downarrow L_{\mathfrak{R}}}$

  $\forall R \in \mathfrak{R}\ \forall z \in L\ \forall Q \in \mathfrak{Q}$

  1) R ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Leftrightarrow \forall u \in R\ \sigma \cdot \langle u, \gamma \rangle \notin \mathbf{\Sigma}\ \&\ \sigma \cdot \langle \Delta \rangle \notin \mathbf{\Sigma}$,

  2) $\sigma \cdot \langle z \rangle \in \mathbf{\Sigma}\ \&\ \exists T \in \mathfrak{R} \cup \mathfrak{Q}\ z \in T\ \&\ \forall u \in T\ \sigma \cdot \langle u, \gamma \rangle \notin \mathbf{\Sigma}\ \&\ \sigma \cdot \langle \Delta \rangle \notin \mathbf{\Sigma}$

  $\Rightarrow \exists P \in \mathfrak{R} \cup \mathfrak{Q}\ z \in P\ \&\ P$ ***safe by*** $\mathbf{\Sigma}$ ***after*** σ,

  3) Q ***safe by*** $\mathbf{\Sigma}$ ***after*** σ

  $\Rightarrow \exists v \in Q\ \sigma \cdot \langle v \rangle \in \mathbf{\Sigma}\ \&\ \forall u \in Q\ \sigma \cdot \langle u, \gamma \rangle \notin \mathbf{\Sigma}\ \&\ \sigma \cdot \langle \Delta \rangle \notin \mathbf{\Sigma}$.

- Под ***F****-спецификацией* будем понимать ***F***-модель с заданным на ней отношением ***safe by***.

· Внешнее действие $z \in L$ *безопасно в спецификационной модели* $\boldsymbol{\Sigma}$ после $\mathfrak{R}$-трассы $\sigma \in \boldsymbol{\Sigma}$, если оно разрешается какой-нибудь кнопкой, безопасной после σ:[10]

z ***safe by*** $\boldsymbol{\Sigma}$ ***after*** $\sigma =_{def} \exists P \in \mathfrak{R} \cup \mathfrak{Q}$ $z \in P$ & P ***safe by*** $\boldsymbol{\Sigma}$ ***after*** σ.

· $\mathfrak{R}$-трассу спецификации $\boldsymbol{\Sigma}$ будем называть безопасной, если спецификация не содержит трассу $\langle\gamma\rangle$, а трасса не заканчивается на дивергенцию и разрушение, и каждый встречающийся в ней символ u (внешнее действие или $\mathfrak{R}$-отказ) безопасен после непосредственно предшествующего ему префикса трассы:

$$\boldsymbol{SafeBy}(\boldsymbol{\Sigma}) =_{def} \{\sigma \in \boldsymbol{\Sigma}^{\downarrow} L_{\mathfrak{R}} \mid \langle\gamma\rangle \notin \boldsymbol{\Sigma} \ \&\ \forall \mu \ \forall u \ (\mu \cdot \langle u \rangle \le \sigma \Rightarrow u \ \textbf{\textit{safe by}}\ \boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)\}.$$

□

Заметим, что, хотя отношение ***safe by*** формально определено на всех $\mathfrak{R}$-трассах спецификации, не заканчивающихся дивергенцией и разрушением, для дальнейшего достаточно считать его определённым только на безопасных трассах. Иными словами, если отношение ***safe by*** определено таким образом, что трасса μ опасна, то нас не будет интересовать, какие кнопки безопасны или опасны после любого продолжения $\mu \cdot \lambda$.

Также заметим, что для любой ***F***-модели можно определить хотя бы одно отношение ***safe by***, удовлетворяющим трём правилам протокола. Например, кнопка объявляется безопасной, если она неразрушающая и, дополнительно для $\mathfrak{Q}$-кнопок, разрешает действие, продолжающее трассу.

---

[10] Заметим, что мы не требуем, чтобы трасса продолжалась в спецификации безопасным внешним действием z.

В общем случае правила протокола не однозначно определяют отношение ***safe by*** даже для безопасных трасс. Однако такая однозначность имеет место в том случае, когда каждое внешнее действие, не разрешаемое $\mathfrak{R}$-кнопками $z \notin \cup\mathfrak{R}$, разрешается только одной $\mathfrak{Q}$-кнопкой $\exists!Q \in \mathfrak{Q}\ z \in Q$. В частности, это имеет место для βγδ-семантики. Действительно, первое правило однозначно определяет безопасность $\mathfrak{R}$-кнопок. Во втором правиле кнопки "T" и "P" совпадают, если действие z, продолжающее трассу, не разрешается $\mathfrak{R}$-кнопками, поскольку только одна $\mathfrak{Q}$-кнопка может разрешать действие z, продолжающее трассу. Вместе с третьим правилом имеем: $\forall z \in L\ \forall Q \in \mathfrak{Q}$

$$Q \textit{ safe by } \Sigma \textit{ after } \sigma \Leftrightarrow \exists v \in Q\ \sigma\cdot\langle v\rangle \in \Sigma\ \&\ \forall u \in Q\ \sigma\cdot\langle u,\gamma\rangle \notin \Sigma\ \&\ \sigma\cdot\langle\Delta\rangle \notin \Sigma.$$

3.6.3. Гипотеза о безопасности

Определение 23: Пусть заданы ***F***-реализация **I** и ***F***-спецификация **Σ**.

- Будем говорить, что реализация **I** *безопасна* для спецификации **Σ**, если, во-первых, в реализации нет разрушения с самого начала, если этого нет в спецификации, и, во-вторых, в реализации любая кнопка, безопасная в спецификации после общей безопасной трассы реализации и спецификации, безопасна после этой трассы в реализации:

$$\mathbf{I} \textit{ safe for } \Sigma =_{\text{def}} (\langle\gamma\rangle \notin \Sigma \Rightarrow \langle\gamma\rangle \notin \mathbf{I})$$
$$\& \forall \sigma \in \textit{SafeBy}(\Sigma) \cap \textit{SafeIn}(\mathbf{I})\ \forall P \in \mathfrak{R}\cup\mathfrak{Q}$$
$$(P \textit{ safe by } \Sigma \textit{ after } \sigma \Rightarrow P \textit{ safe in } \mathbf{I} \textit{ after } \sigma).$$

- Множество реализаций, безопасных для данной спецификации, обозначим:

$$\textit{safe}\mathfrak{I}(\Sigma) =_{\text{def}} \{\mathbf{I} \in \textit{FMODEL}(L) \mid \mathbf{I} \textit{ safe for } \Sigma\}.$$

□

Безопасность внешнего действия после трассы означает, что оно разрешается кнопкой, безопасной после этой трассы. Поэтому гипотезу о безопасности можно сформулировать в следующем эквивалентном виде:

$\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$ $=_{\text{def}}$ $(\langle\gamma\rangle\notin\mathbf{\Sigma} \Rightarrow \langle\gamma\rangle\notin\mathbf{I})$

$\&\ \forall\sigma\in\boldsymbol{SafeBy}(\mathbf{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I})\ \ \forall u\in L_{\mathfrak{R}\cup\mathfrak{Q}}$

$(u$ ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Rightarrow u$ ***safe in*** $\mathbf{I}$ ***after*** $\sigma)$.

Лемма 6: В определении отношения ***safe for*** вместо $\sigma\in\boldsymbol{SafeIn}(\mathbf{I})$ достаточно потребовать $\sigma\in\mathbf{I}$, то есть

$\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$ $=$ $(\langle\gamma\rangle\notin\mathbf{\Sigma} \Rightarrow \langle\gamma\rangle\notin\mathbf{I})\ \&\ \forall\sigma\in\boldsymbol{SafeBy}(\mathbf{\Sigma})\cap\mathbf{I}\ \ \forall P\in\mathfrak{R}\cup\mathfrak{Q}$

$(P$ ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Rightarrow P$ ***safe in*** $\mathbf{I}$ ***after*** $\sigma)$.

□451

Лемма 7: Если безопасная трасса спецификации принадлежит безопасной реализации, то она безопасна в ней.

□452

3.6.4. Безопасная конформность

Определение 24: Пусть задана ***F***-модель $\mathbf{T}$. Обозначим множество внешних действий и $\mathfrak{R}$-отказов, наблюдаемых в модели T после $\mathfrak{R}$-трассы σ и нажатии кнопки "P":

$\boldsymbol{obs}(\sigma, P, \mathbf{T}) =_{\text{def}} \{\ u \mid \sigma\cdot\langle u\rangle\in\mathbf{T}\ \&\ (u\in P \vee u=P\ \&\ P\in\mathfrak{R})\ \}.$

□

Определение 25: Пусть заданы ***F***-реализация $\mathbf{I}$ и ***F***-спецификация $\mathbf{\Sigma}$.

- Будем говорить, что реализация $\mathbf{I}$ *безопасно конформна* (или просто *конформна*) спецификации $\mathbf{\Sigma}$, если она безопасна и выполнено *тестируемое условие*: любое наблюдение, возможное в реализации в ответ на нажатие безопасной (в спецификации) кнопки, разрешается спецификацией:

  $\mathbf{I}$ ***saco*** $\mathbf{\Sigma}$ $=_{\text{def}}$ $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$ $\&$

$\forall\sigma\in\mathit{SafeBy}(\Sigma)\cap\mathit{SafeIn}(\mathbf{I})\ \ \forall \mathtt{P}\ \ \mathit{safe\ by}\ \ \Sigma\ \ \mathit{after}\ \ \sigma$

$\mathit{obs}(\sigma,\mathtt{P},\mathbf{I})\subseteq\mathit{obs}(\sigma,\mathtt{P},\Sigma)$.

· Множество реализаций, конформных спецификации Σ, обозначим:

$\mathfrak{I}(\Sigma) =_{\text{def}} \{\mathbf{I}\in\mathit{FMODEL}(\mathtt{L})\,|\,\mathbf{I}\ \mathit{saco}\ \Sigma\}$.

· Обозначим: $\Sigma_1 \sim \Sigma_2 =_{\text{def}} \Sigma_1\ \mathit{saco}\ \Sigma_2$` & $\Sigma_2\ \mathit{saco}\ \Sigma_1$.

□

Лемма 8: В определении отношения ***saco*** вместо $\sigma\in\mathit{SafeIn}(\mathbf{I})$ достаточно потребовать $\sigma\in\mathbf{I}$, то есть

$\mathbf{I}\ \mathit{saco}\ \Sigma =$ $\mathbf{I}\ \mathit{safe\ for}\ \Sigma$ & $\forall\sigma\in\mathit{SafeBy}(\Sigma)\cap\mathbf{I}\ \forall\mathtt{P}\ \mathit{safe\ by}\ \Sigma\ \mathit{after}\ \sigma$

$\mathit{obs}(\sigma,\mathtt{P},\mathbf{I})\subseteq\mathit{obs}(\sigma,\mathtt{P},\Sigma)$.

□452

Условие конформности есть конъюнкция двух условий: безопасность реализации и тестируемое условие. Безопасность реализации – это гипотеза о реализации, то есть предусловие тестирования, оно предполагается выполненным и не проверяется при тестировании. Тестируемое условие – это и есть то условие, которое проверяется при тестировании.

Там, где по контексту неясно, в какой $\mathfrak{R}/\mathfrak{Q}$-семантике рассматриваются отношения и множества, мы будем указывать семантику в нижнем индексе и писать, соответственно: $\mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{in}$, $\mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{by}$, $\mathit{SafeIn}_{\mathfrak{R}/\mathfrak{Q}}$, $\mathit{SafeBy}_{\mathfrak{R}/\mathfrak{Q}}$, $\mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{for}$, $\mathit{safe}\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}$, $\mathit{obs}_{\mathfrak{R}/\mathfrak{Q}}$, $\mathit{saco}_{\mathfrak{R}/\mathfrak{Q}}$, $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}$. и $\sim_{\mathfrak{R}/\mathfrak{Q}}$.

В качестве примера можно рассмотреть γδ-семантику для реактивных систем. Алфавит внешних действий разделяется на стимулы и реакции: $\mathtt{L}=\{\mathtt{?x}\,|\,\mathtt{?x}\in\mathtt{L}\}\cup\{\mathtt{!y}\,|\,\mathtt{!y}\in\mathtt{L}\}$. Для приёма реакций от реализации имеется единственная $\mathfrak{R}$-кнопка “δ”, разрешающая передачу любой реакции; соответствующий наблюдаемый отказ называется стационарностью и обозначается символом $\delta=\{\mathtt{!y}\,|\,\mathtt{!y}\in\mathtt{L}\}$. Для передачи каждого стимула $\mathtt{?x}\in\mathtt{L}$ имеется отдельная $\mathfrak{Q}$-кнопка “$\{\mathtt{?x}\}$”; соответствующий ненаблюдаемый отказ

называется блокировкой стимула `?x`. Отношение ***saco*** для этой семантики почти совпадает с отношением ***ioco***. Отличие только в том, что домен отношения ***ioco*** уже: в спецификациях и реализациях нет дивергенции и разрушения (δ-семантика), а в реализации, кроме того, нет блокировок стимулов. На этом общем домене отношения совпадают. Если блокировки наблюдаемы (𝔑-кнопки), будем говорить о βγδ-семантике или, если разрушение и дивергенция запрещены, о βδ-семантике. Отношения ***saco*** в соответствующих семантиках будем обозначать $\boldsymbol{ioco}_{\delta}$, $\boldsymbol{ioco}_{\gamma\delta}$, $\boldsymbol{ioco}_{\beta\delta}$ и $\boldsymbol{ioco}_{\beta\gamma\delta}$ [21-23,25-27].

3.7. Генерация тестов

Структура раздела:

1. Тестовые трассы
2. Вердикт
3. Конечность времени выполнения теста
4. Управляемые тесты
5. Определение теста и тестового набора
6. Значимые, исчерпывающие и полные наборы тестов
7. Строгие тесты
8. Полный набор примитивных тестов
9. **Ошибка! Источник ссылки не найден.**
10. Алгоритмизация

В терминах машины тестирования тест можно понимать как инструкцию оператору, которая указывает ему, какую кнопку нужно нажимать в тот или иной момент времени, когда заканчивать тестирование и какой выносить вердикт после окончания тестирования. Все эти указания зависят от предшествующей последовательности нажатия кнопок и наблюдений поведения машины.

Для машины без приоритетов в этой последовательности важны лишь наблюдения (внешние действия и отказы). Таким образом выбор нажимаемой кнопки, окончание тестирования и вынесение вердикта определяется полученной к данному моменту времени трассой наблюдений. Поэтому в трассовой модели тест – это просто набор $\mathfrak{R}$-трасс, которые мы хотим проверить этим тестом. Тест всегда является деревом, поскольку получение любой трассы означает также получение всех её префиксов в более ранние моменты времени. Для каждой немаксимальной трассы теста нужно указать, какие кнопки можно нажимать после получения этой трассы. Тест заканчивается после получения максимальной трассы, которая определяет, какой нужно выносить вердикт.

Мы налагаем на тесты два ограничения: 1) тестирование должно быть безопасным и 2) любой прогон любого теста должен заканчиваться за конечное время с вынесением вердикта.

При безопасном тестировании оператор может нажимать кнопку только в том случае, когда он получил трассу, которая безопасна в спецификации, и кнопка безопасна в спецификации после этой трассы. В этом случае гипотеза о безопасности гарантирует, что дивергенции и разрушения не будет (в реализации), и через конечное время будет наблюдаться внешнее действие или $\mathfrak{R}$-отказ. Поэтому дополнительное требование конечности времени прогона теста означает, что тест не должен требовать от оператора бесконечного числа нажатий кнопок, а также не ставит оператора «в тупик», когда дальнейшие действия оператора не определяются тестом.

3.7.1. Тестовые трассы

Если в спецификации есть трасса $\langle\gamma\rangle$, гипотеза о безопасности не гарантирует отсутствие разрушения реализации сразу после включения машины ещё до нажатия каких бы то ни было кнопок. Однако такую реализацию нам и не нужно тестировать, поскольку, если в спецификации есть

трасса $\langle\gamma\rangle$, то в ней нет безопасных трасс, и ей конформна любая реализация. Тестирование реализации имеет смысл только для спецификации, в которой нет трассы $\langle\gamma\rangle$.

Если в спецификации нет трассы $\langle\gamma\rangle$, то при безопасном тестировании мы можем включать машину тестирования и сразу же наблюдаем пустую трассу, которая безопасна в спецификации и в безопасной реализации. Пусть в какой-то момент времени мы получили трассу σ, безопасную в спецификации и в безопасной реализации. После нажатия кнопки "P", которая безопасна в спецификации после трассы σ, через конечное время наблюдается некоторый символ u, продолжающий трассу σ в реализации: либо действие $u{=}z{\in}P$, либо $\mathfrak{R}$-отказ $u{=}P$, если $P{\in}\mathfrak{R}$. Далее нужно проверить, есть ли трасса $\sigma{\cdot}\langle u\rangle$ в спецификации или нет. В первом случае мы можем как продолжить тестирование, так и закончить его, а во втором случае тестирование обязательно заканчивается, поскольку безопасность дальнейшего тестирования не гарантируется гипотезой о безопасности.

Таким образом, любая наблюдаемая при безопасном тестировании трасса $\sigma{\cdot}\langle u\rangle$ либо безопасна в спецификации, либо продолжает безопасную трассу σ безопасным после неё символом u. Такие трассы мы будем называть *тестовыми трассами*. Мы можем считать, что тест – это дерево тестовых трасс. Заметим, что все немаксимальные в тесте тестовые трассы должны быть безопасными трассами спецификации. Максимальные трассы теста либо безопасны в спецификации, либо отсутствуют в ней.

Определение 26: Пусть задана ***F***-спецификация $\mathbf{\Sigma}$. *Тестовой трассой* будем называть такую трассу σ, которая является безопасной трассой или продолжает безопасную трассу безопасным символом:

$$\exists\mu{\in}\textbf{\textit{SafeBy}}(\mathbf{\Sigma})\ \ \sigma{=}\mu\ \vee\ \exists u{\in}L_{\mathfrak{R}}\ u\ \textbf{\textit{safe by}}\ \mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu\ \&\ \sigma{=}\mu{\cdot}\langle u\rangle.$$

Множество тестовых трасс обозначим $\boldsymbol{tt}(\boldsymbol{\Sigma})$.

□

Заметим, что множество тестовых трасс спецификации зависит как от $\mathfrak{R}/\mathfrak{Q}$-семантики, так и от выбранного протокола взаимодействия (отношения ***safe by***).

Лемма 9: Безопасные трассы спецификации – это те её тестовые трассы, которые ей принадлежат: $\boldsymbol{tt}(\boldsymbol{\Sigma}) \cap \boldsymbol{\Sigma} = \boldsymbol{SafeBy}(\boldsymbol{\Sigma})$.

□453

Заметим, что в общем случае по безопасным трассам спецификации нельзя определить её тестовые трассы. Пример приведён на Рис.33.

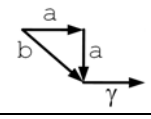
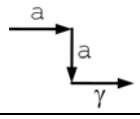
По безопасным трассам не восстанавливаются тестовые трассы			
Порождающий граф спецификации			L={a,b} $\mathfrak{R}$={{a},{a,b}} $\mathfrak{Q}$=∅
Безопасные трассы	$\epsilon, \langle a \rangle$		
Тестовые трассы	$\epsilon, \langle \{a\} \rangle, \langle a \rangle$	$\epsilon, \langle \{a\} \rangle, \langle a \rangle, \langle \{a,b\} \rangle, \langle b \rangle$	

Рис.33.

В то же время, если каждое внешнее действие разрешается только одной кнопкой (любые два кнопочных множества имеют пустое пересечение), то по безопасным трассам спецификации можно восстановить её тестовые трассы. В частности, это имеет место для βγδ-семантики. Для восстановления тестовых трасс нужно, чтобы можно было определить все кнопки, безопасные после безопасной трассы μ по множеству всех безопасных трасс (а не по множеству всех трасс) спецификации. В силу $\mathfrak{R}$-конвергентности спецификации, $\mathfrak{R}$-отказ P безопасен после трассы μ тогда и только тогда, когда безопасна трасса $\mu \cdot \langle P \rangle$ или $\mu \cdot \langle z \rangle$ для некоторого действия $z \in P$, что можно определить по множеству безопасных трасс. Также $\mathfrak{Q}$-отказ P безопасен после трассы μ

тогда и только тогда, когда безопасна трасса $\mu\cdot\langle z\rangle$ для некоторого действия $z\in P$, что также можно определить по множеству безопасных трасс.

3.7.2. Вердикт

Оператор в конце своей работы должен вынести вердикт ***pass***, означающий, что реализация «прошла» тест, или ***fail*** в противном случае. Заметим, что вердикты ***pass*** и ***fail***, вообще говоря, не связаны с конформностью или неконформностью реализации спецификации. Эту связь мы установим ниже, когда будем определять значимые, исчерпывающие и полные наборы тестов.

Когда оператор заканчивает свою работу? В трассовой модели это может определяться только трассой, наблюдаемой к этому моменту времени. В общем случае можно было бы назначить вердикт произвольным трассам теста и даже одной трассе назначить оба вердикта ***pass*** и ***fail***. Тогда оператору предоставляется выбор: закончить тест с вынесением вердикта или, если это немаксимальная трасса, продолжать тестирование; если тест заканчивается, а трассе назначены оба вердикта, выбирается один из них. Понятно, что такой недетерминизм теста не увеличивает мощности тестирования. Чтобы проверить все *способы исполнения оператором* недетерминированного теста, мы должны прогонять его несколько раз для каждого *варианта* погодных условий, детерминирующего выбор *поведения реализации*: каждый такой прогон соответствует одному способу исполнения теста оператором. Это эквивалентно наличию нескольких детерминированных тестов, каждый из которых прогоняется один раз для каждого варианта погодных условий.

Поэтому будем считать, что окончанию прогона теста соответствует максимальная трасса в дереве тестовых трасс, и для каждой максимальной трассы определён один вердикт (а не оба). Оператор заканчивает работу тогда и только тогда, когда получена максимальная трасса теста, после чего выносится вердикт, назначенный этой максимальной трассе.

Определение 27: Пусть заданы ***F***-спецификация Σ и дерево тестовых трасс $T \subseteq tt(\Sigma)$.

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

□

3.7.3. Конечность времени выполнения теста

Выполнение теста есть последовательность шагов, каждый из которых представляет собой следующую последовательность элементарных шагов:

a) Принятие решения:
   - после наблюдения максимальной трассы теста завершение прогона теста и вынесение вердикта, назначенного этой трассе;
   - после наблюдения немаксимальной трассы теста выбор безопасной кнопки, которую нужно нажимать для продолжения тестирования.
b) Нажатие кнопки, моделирующее тестовое воздействие на реализацию.
c) Ожидание ответа после нажатия кнопки "P": внешнего действия $z \in P$ или отказа P, если $P \in \mathfrak{R}$.
d) Наблюдение ответа и добавление его в конец наблюдаемой трассы.

Время выполнения теста конечно, если:

1) конечна последовательность шагов,
2) оператор не попадает в тупик: принятие решения всегда возможно,
3) ответ на нажимаемую кнопку обязательно поступит,
4) время выполнения каждого элементарного шага конечно, причём принятие решения конечно, если оно возможно по п.2, а время ожидания ответа конечно, если ответ поступит в соответствии с п.3.

Мы будем предполагать, что условие 4) выполнено. Выполнение шагов a), b) и d) определяется конечной скоростью работы оператора. Выполнение шага c) определяется конечным временем выполнения каждого внешнего или

внутреннего действия машины, а также конечным временем обнаружения отказа. Условие 3) гарантируется безопасностью тестирования. Выполнение условий 1) и 2) определяются устройством самого теста как дерева тестовых трасс. Сформулируем соответствующие ограничения на тест.

1) Конечность последовательности шагов при любом прогоне теста означает, что каждая трасса теста не может продолжаться в тесте сколь угодно долго и рано или поздно заканчивается вердиктом ***pass*** или ***fail***. Иными словами, в тестовом дереве не должно быть бесконечно возрастающей последовательности трасс, то есть тестовое дерево должно быть нётеровым.

   Заметим, что из нётеровости тестового дерева вовсе не следует ни его конечность (как множества трасс), ни ограниченность длин его трасс (см. Рис.34).

Нётеровость, конечность и ограниченность дерева

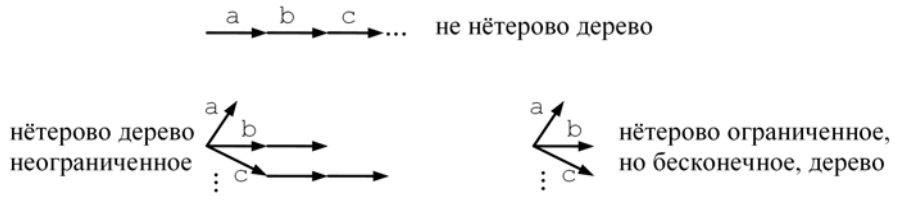


Рис.34.

Ограниченность теста означала бы не просто конечность времени выполнения: ещё до начала прогона 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 внешних действий.

2) Оператор не попадает в тупик, если после наблюдения каждой трассы, которая возможна при тестировании, тест определяет дальнейшее поведение

оператора. После наблюдения максимальной трассы поведение однозначно: завершение прогона теста и вынесение вердикта, назначенного этой трассе. Для немаксимальной тестовой трассы σ это означает, что с ней ассоциирована хотя бы одна кнопка, которая 1) безопасна после трассы σ и, следовательно, её можно нажимать после σ, и 2) любое действие или отказ, которые мы можем наблюдать после нажатия такой кнопки, продолжает трассу σ в тесте. Кнопку, удовлетворяющую этим двум условиям, будем называть допустимой после трассы. Понятно, что мы можем нажимать любую кнопку, допустимую после трассы. Поэтому нам нет нужды дополнительно указывать кнопки, ассоциированные с немаксимальной трассой, аналогично тому, как мы назначали вердикт каждой максимальной трассе. Наличие двух таких кнопок означает лишь недетерминизм в поведении оператора, но не тупик.

Определение 28: Пусть заданы ***F***-спецификация $\mathbf{\Sigma}$ и дерево тестовых трасс $\mathbf{T}\subseteq \boldsymbol{tt}(\mathbf{\Sigma})$.

- Будем говорить, что кнопка "P" *допустима* после немаксимальной тестовой трассы $\sigma\in\mathbf{T}\setminus\boldsymbol{max}(\mathbf{T})$, если она безопасна (в спецификации) после этой трассы, а трасса в тесте продолжается каждым действием, которое разрешается этой кнопкой и после которого трасса остаётся согласованной, а также отказом P, если "P" это $\mathfrak{R}$-кнопка:

  P допустима после σ $=_{\text{def}}$ $\forall z\in P$ $(\sigma\cdot\langle z\rangle$ согласована $\Rightarrow$ $\sigma\cdot\langle z\rangle\in\mathbf{T})$

  $\&$ $(P\in\mathfrak{R} \Rightarrow \sigma\cdot\langle P\rangle\in\mathbf{T})$.

- Будем говорить, что дерево тестовых трасс $\mathbf{T}\subseteq\boldsymbol{tt}(\mathbf{\Sigma})$ удовлетворяет условию отсутствия тупика, если после каждой его немаксимальной трассы имеется допустимая кнопка:

  $\forall\sigma\in\mathbf{T}\setminus\boldsymbol{max}(\mathbf{T})$ $\exists P\in\mathfrak{R}\cup\mathfrak{Q}$ P допустима после σ.

□

Следует отметить, что условие отсутствия тупика ничего не говорит о том, какими ещё внешними действиями (или отказами) может продолжаться немаксимальная трасса теста при условии, что она остаётся тестовой трассой, помимо тех, что принадлежат допустимому отказу (или совпадают с допустимым $\mathfrak{R}$-отказом).

Формально, продолжение трассы внешним действием z или отказом Q может быть не связано с допустимыми кнопками, то есть возможно, что для каждой кнопки "P", допустимой после трассы, $z \notin P$ или $Q \neq P$. Понятно, что все такие продолжения можно удалить из теста без изменения мощности тестирования.

Другие «лишние» продолжения тестовой трассы связаны с условием согласованности. Это условие требует: если трасса σ продолжается внешним действием z, то трасса $\sigma \cdot \langle z \rangle$ должна быть согласована. Несогласованные трассы могут быть в тесте, но они «лишние», так как не могут наблюдаться ни в какой реализации. Если в условии допустимости кнопки заменить обе импликации "$\Rightarrow$" на эквивалентность "$\Leftrightarrow$", то такой тест не будет содержать несогласованных трасс.

Ниже мы для того, чтобы избежать «излишнего» недетерминизма в тесте, связанного с выбором той или иной кнопки после трассы, потребуем, чтобы выбор нажимаемой кнопки определялся по возможности однозначно, то есть некоторые «лишние» продолжения немаксимальных трасс запретим. В то же время для упрощения генерации тестов разрешим некоторые другие «лишние» (несогласованные) продолжения немаксимальных трасс.

3.7.4. Управляемые тесты

Если после немаксимальной трассы теста можно выбрать разные кнопки, которые можно нажимать при тестировании, то это создаёт ненужный недетерминизм в поведении оператора, который не увеличивает мощности тестирования. Как и в случае с вердиктом на немаксимальных трассах, чтобы

проверить все способы выбора допустимых кнопок, мы должны прогонять тест несколько раз для каждого варианта погодных условий, детерминирующего выбор поведения реализации. Это эквивалентно наличию нескольких тестов с детерминированным выбором допустимых кнопок, каждый из которых прогоняется один раз для каждого варианта погодных условий.

Итак, мы хотим, чтобы выбираемая кнопка однозначно определялась теми действиями и отказами, которыми трасса σ продолжается в тесте. Для этого нужно, чтобы немаксимальная трасса σ продолжалась в тесте $\mathbf{T}$ либо множеством действий $P \in \mathfrak{Q}$, либо множеством действий $P \in \mathfrak{R}$ и отказом P, где P допустимо после σ. После трассы σ мы будем всегда нажимать только кнопку "P". Такие тесты будем называть *управляемыми*. Любая другая кнопка "Q", допустимая после σ, является $\mathfrak{Q}$-кнопкой и $Q \subset P$. Очевидно, нажатие такой кнопки "Q" гарантированно не увеличивает мощность тестирования данным тестом.

Определение 29: Пусть заданы ***F***-спецификация $\mathbf{\Sigma}$ и дерево тестовых трасс $\mathbf{T} \subseteq \boldsymbol{tt}(\mathbf{\Sigma})$.

- Будем говорить, что трасса $\sigma \in \mathbf{T}$ является *P-трассой*, где $P \in \mathfrak{R} \cup \mathfrak{Q}$, если выполнено следующее условие:

  $P \in \mathfrak{Q}$ & $\boldsymbol{head}(\mathbf{T}\ \boldsymbol{after}\ \sigma) = P$ $\vee$ $P \in \mathfrak{R}$ & $\boldsymbol{head}(\mathbf{T}\ \boldsymbol{after}\ \sigma) = P \cup \{P\}$.

- Будем говорить, что дерево $\mathbf{T}$ *управляемое*, если каждая его немаксимальная трасса является P-трассой для некоторого $P \in \mathfrak{R} \cup \mathfrak{Q}$:

  $\forall \sigma \in \mathbf{T} \setminus \boldsymbol{max}(\mathbf{T})$ $\exists P \in \mathfrak{R} \cup \mathfrak{Q}$ σ P-трасса.

□

Очевидно, что в управляемом дереве тестовых трасс для каждой немаксимальной P-трассы σ кнопка "P" допустима после σ. Отметим, что управляемые тесты могут содержать «лишние» несогласованные трассы, которые никогда не будут наблюдаться при тестировании. Такие трассы могут

быть только максимальными. Удаление таких трасс может сделать выбор кнопки после немаксимальной трассы неоднозначным, что практически неудобно.

Пусть, например, L={a,b,c}, $\mathfrak{R}$={{a},{b}}, $\mathfrak{Q}$={{a,c},{b,c}} и в спецификации есть безопасная трасса ⟨{a},{b},c⟩. Для проверки действия c после трассы ⟨{a},{b}⟩ мы можем нажимать кнопку “{a,c}” или кнопку “{b,c}”. Пусть по протоколу взаимодействия обе эти кнопки объявлены безопасными после трассы ⟨{a},{b}⟩. В управляемом тесте будет трасса ⟨{a},{b},c⟩ и обязательно одна из несогласованных трасс ⟨{a},{b},a⟩ или ⟨{a},{b},b⟩ для кнопок “{a,c}” или “{b,c}”, соответственно. Удаление такой несогласованной трассы делает выбор кнопки “{a,c}” или “{b,c}” неоднозначным.

Можно также отметить, что если в $\mathfrak{R}$/$\mathfrak{Q}$-семантике каждое внешнее действие разрешается только одной кнопкой (кнопочные множества не пересекаются), то каждое действие или $\mathfrak{R}$-отказ, которыми продолжается трасса в управляемом тесте, однозначно определяет кнопку “P” после этой трассы (она является P-трассой только для одного отказа P). В этом случае управляемый тест однозначно определяется подмножеством его трасс, безопасных в спецификации. Несогласованность трассы управляемого теста может означать только, что между двумя нажатиями одной и той же $\mathfrak{R}$-кнопки “P” наблюдаются только отказы, и поэтому после второго отказа P любое внешнее действие z∈P делает трассу несогласованной. В этом случае можно удалить повторные отказы и, тем самым, несогласованные трассы из управляемого теста. В частности, это имеет место для βγδ-семантики [27].

3.7.5. Определение теста и тестового набора

Определение 30: Пусть задана ***F***-спецификация Σ.

- *Тестовым примером (test case)* или просто *тестом* для спецификации Σ будем называть управляемое нётерово дерево тестовых трасс с заданным вердиктом.
- Множество всех тестовых примеров обозначим $\boldsymbol{Ttests}(\Sigma, \mathfrak{R}, \mathfrak{Q})$.
- *Набор тестов или тестовый набор (test suite)* TT – это множество тестовых примеров: $\mathrm{TT} \subseteq \boldsymbol{Ttests}(\Sigma, \mathfrak{R}, \mathfrak{Q})$.

□

Заметим, что трассовый тест, вообще говоря, не является трассовой моделью. В нём могут быть несогласованные трассы, но, даже если их нет, всегда есть максимальные трассы, которые неконвергентны. Исключение составляет только вырожденный случай, когда нет $\mathfrak{R}$-кнопок: любое дерево допустимых трасс является трассовой моделью, а, поскольку в тесте нет дивергенции и разрушения, оно является таким деревом.

3.7.6. Значимые, исчерпывающие и полные наборы тестов

Реализация *проходит* тест, если при любом прогоне этого теста (при любых погодных условиях) выносится вердикт ***pass***. Это означает, что реализация *проходит* тест тогда и только тогда, когда 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 31: Пусть задана ***F***-спецификация $\mathbf{\Sigma}$. Тогда для безопасной реализации $\mathbf{I}\in\boldsymbol{safe}\mathfrak{I}(\mathbf{\Sigma})$, теста $\mathbf{T}\in\boldsymbol{Ttests}(\mathbf{\Sigma},\mathfrak{R},\mathfrak{Q})$ и тестового набора $\mathbf{TT}\subseteq\boldsymbol{Ttests}(\mathbf{\Sigma},\mathfrak{R},\mathfrak{Q})$ определим:

- $\mathbf{I}$ ***passes*** $\mathbf{T}$ $=_{\text{def}}$ $\mathbf{I}\cap\boldsymbol{fail}(\mathbf{T})=\varnothing$;
- $\mathbf{I}$ ***passes*** $\mathbf{TT}$ $=_{\text{def}}$ $\forall\mathbf{T}\in\mathbf{TT}$ $\mathbf{I}$ ***passes*** $\mathbf{T}$;
- $\mathbf{TT}$ ***sound*** $\mathbf{\Sigma}$ $=_{\text{def}}$ $\forall\mathbf{I}\in\boldsymbol{safe}\mathfrak{I}(\mathbf{\Sigma})$ $(\mathbf{I}$ ***saco*** $\mathbf{\Sigma}\Rightarrow\mathbf{I}$ ***passes*** $\mathbf{TT})$;
- $\mathbf{TT}$ ***exhaustive*** $\mathbf{\Sigma}$ $=_{\text{def}}$ $\forall\mathbf{I}\in\boldsymbol{safe}\mathfrak{I}(\mathbf{\Sigma})$ $(\mathbf{I}$ ***saco*** $\mathbf{\Sigma}\Leftarrow\mathbf{I}$ ***passes*** $\mathbf{TT})$;
- $\mathbf{TT}$ ***complete*** $\mathbf{\Sigma}$ $=_{\text{def}}$ $\forall\mathbf{I}\in\boldsymbol{safe}\mathfrak{I}(\mathbf{\Sigma})$ $(\mathbf{I}$ ***saco*** $\mathbf{\Sigma}\Leftrightarrow\mathbf{I}$ ***passes*** $\mathbf{TT})$.

□

Задача генерации тестов заключается в построении в заданной $\mathfrak{R}/\mathfrak{Q}$-семантике полного набора тестов по заданной спецификации $\mathbf{\Sigma}$ для заданного протокола взаимодействия (отношения ***safe by***).

3.7.7. Строгие тесты

Тестовая трасса, отсутствующая в спецификации, может быть только максимальной непустой трассой теста. Поскольку такая трасса σ является продолжением безопасной в спецификации трассы μ символом u, безопасным после трассы μ в спецификации, но отсутствующим в самой спецификации, трасса $\sigma=\mu\cdot\langle u\rangle$ не может быть ни в какой конформной реализации. Хотя значимый тест не обязан назначать такой трассе σ вердикт ***fail***, такой тест остаётся значимым, если мы переопределим $\boldsymbol{verdict}(\sigma):=\boldsymbol{fail}$. Очевидно, такой модифицированный тест находит не меньшее число ошибок, чем исходный (и не находит «ложных» ошибок, поскольку это значимый тест). Таким образом, мы можем ограничиться тестами, для которых верна импликация: $\forall\sigma\in\mathbf{T}$ $(\sigma\notin\mathbf{\Sigma}\Rightarrow\sigma\in\boldsymbol{fail}(\mathbf{T}))$.

В то же время в значимом тесте могут быть ***fail***-трассы, которые являются трассами спецификации. Значимость теста означает только, что такая ***fail***-трасса не может встречаться в конформных реализациях. Иными словами, в спецификации могут быть «лишние» трассы, которые не встречаются ни в одной конформной реализации. Пример такой спецификации мы рассматривали в предыдущей главе (Рис.15). Здесь можно лишь отметить, что такой несколько неожиданный эффект возникает в $\mathfrak{R}/\mathfrak{Q}$-семантике только в том случае, когда имеются кнопки с ненаблюдаемым отказом, то есть когда $\mathfrak{Q}\neq\varnothing$ (см. ниже Теорема 17: в главе 5).

Тем не менее, мы покажем, что для полноты тестирования достаточно ограничиться тестами, в которых выполнена обратная импликация: $\forall\sigma\in\mathbf{T}\ (\sigma\notin\mathbf{\Sigma} \Leftarrow \sigma\in\boldsymbol{fail}(\mathbf{T}))$. В общем мы приходим к тестам, для которых имеет место эквивалентность $\forall\sigma\in\mathbf{T}\ (\sigma\notin\mathbf{\Sigma} \Leftrightarrow \sigma\in\boldsymbol{fail}(\mathbf{T}))$, и которые мы будем называть *строгими тестами*.

Определение 32: Пусть задана ***F***-спецификация $\mathbf{\Sigma}$.

- Тест $\mathbf{T}\in\boldsymbol{Ttests}(\mathbf{\Sigma},\mathfrak{R},\mathfrak{Q})$ будем называть *строгим*, если выполняется следующее условие: $\forall\sigma\in\mathbf{T}\ (\sigma\notin\mathbf{\Sigma} \Leftrightarrow \sigma\in\boldsymbol{fail}(\mathbf{T}))$.
- Тестовый набор будем называть строгим, если все его тесты строгие.

□

По Лемме 9, для тестовой трассы σ условие $\sigma\in\mathbf{\Sigma}$ эквивалентно $\sigma\in\boldsymbol{SafeBy}(\mathbf{\Sigma})$. Поэтому строгость теста – это жёсткое правило назначения вердикта максимальным трассам теста: вердикт ***pass*** получают безопасные трассы спецификации, а вердикт ***fail*** – трассы, отсутствующие в спецификации. Множество тестовых трасс спецификации однозначно определяет класс безопасных реализаций, а вместе с подмножеством безопасных трасс они однозначно определяют класс конформных реализаций.

Теорема 7: Строгий тестовый набор $\mathbf{TT}$, покрывающий все тестовые трассы спецификации $\boldsymbol{tt}(\boldsymbol{\Sigma}) \subseteq \cup\mathbf{TT}$, является полным.

□453

3.7.8. Полный набор примитивных тестов

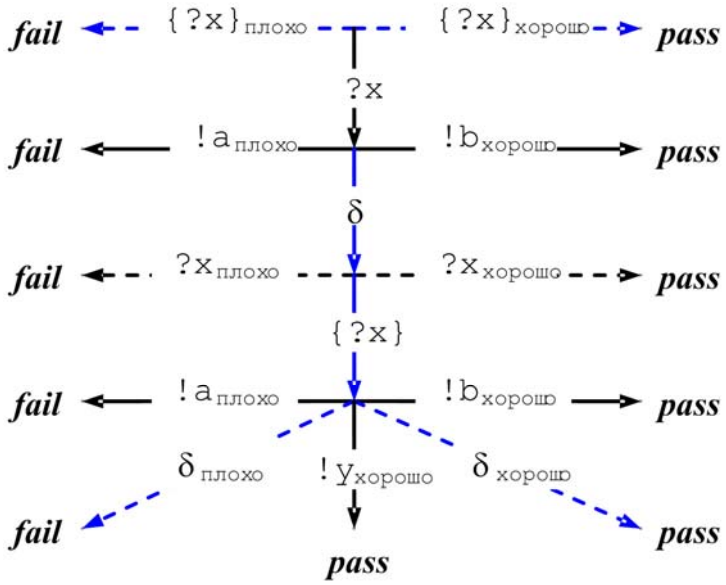
В качестве примера управляемого теста рассмотрим *примитивный* тест[11], который строится по произвольной выделенной тестовой трассе σ (Рис.35). Трасса σ объявляется максимальной трассой теста. Для каждого её префикса μ, продолжаемого в трассе символом u, то есть $\mu\cdot\langle u\rangle \le \sigma$, выбирается любая кнопка "P", безопасная в спецификации после μ и такая, что $u \in P$ или $u = P$ (для $P \in \mathfrak{R}$). Существование такой кнопки гарантируется тем, что σ – это тестовая трасса спецификации. Далее для каждого $t \neq u$ такого, что $t \in P$ или $t = P$ (для $P \in \mathfrak{R}$), добавляется максимальная трасса $\mu\cdot\langle t\rangle$. Наконец, добавляем все префиксы максимальных трасс и устанавливаем такие вердикты максимальных трасс, чтобы получился строгий тест.

Определение 33: Пусть задана $\boldsymbol{F}$-спецификация $\boldsymbol{\Sigma}$. Будем говорить, что тест $\mathbf{T}$ является *примитивным* тестом для тестовой трассы $\sigma \in \boldsymbol{tt}(\boldsymbol{\Sigma})$, если он строгий, управляемый и каждая его максимальная трасса либо совпадает с σ, либо является ответвлением от σ на один символ: $\boldsymbol{max}(\mathbf{T}) = \{\sigma\} \cup (\mathbf{T} \setminus \boldsymbol{Tree}(\sigma))$.

□

[11] Название «примитивный» выбрано потому, что другие тесты можно представить в виде объединения примитивных тестов.

Генерация порождающего графа примитивного теста для тестовой трассы $\delta=\langle ?x, \delta, \{?x\}, !y\rangle$ в βγδ-семантике



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.35.

В общем случае примитивный тест строится по безопасной трассе спецификации неоднозначно: после некоторых префиксов могут выбираться те или иные безопасные кнопки, разрешающие продолжающий префикс внешнее действие. В то же время, как отмечалось выше, если в $\mathfrak{R}/\mathfrak{Q}$-семантике каждое внешнее действие разрешается только одной кнопкой (в частности, в βγδ-семантике), то управляемый тест однозначно определяется подмножеством его трасс, безопасных в спецификации. В частности, примитивный тест однозначно строится по одной безопасной трассе спецификации. Следующее утверждение можно считать утверждением о существовании полных тестовых наборов.

Теорема 8: Множество примитивных тестов для всех тестовых трасс спецификации является полным набором ограниченных строгих управляемых тестов.

□454

3.7.9. Оптимизация

На самом деле для полноты тестирования нам часто вовсе не нужно покрывать все тестовые трассы спецификации или иметь каждую такую трассу в качестве максимальной трассы. Это даёт возможность различного рода оптимизаций:

1. Удаление несогласованных тестовых трасс.

Такие трассы не могут наблюдаться при тестировании любой реализации, поэтому их можно удалить из любого теста. Заметим, что несогласованная тестовая трасса всегда является максимальной трассой теста и имеет вид $\mu\cdot\langle \mathtt{P}\rangle\cdot\rho\cdot\langle \mathtt{z}\rangle$, где $\mathtt{P}\in\mathfrak{R}$, $\rho\in\mathfrak{R}^*$ и $\mathtt{z}\in\mathtt{P}$.

Также ещё раз отметим, что управляемые тесты часто содержат несогласованные трассы, но это полезно на практике для однозначности выбора кнопки после немаксимальных трасс теста. Если мы хотим удалить несогласованные трассы, но сохранить детерминизм теста, нам нужно каким-то дополнительным способом указывать допустимую кнопку, которую оператор должен нажимать после немаксимальной тестовой трассы (фактически, использовать тестовые истории).

2. Удаление «лишних» отказов.

Повторные отказы (одинаковые отказы в одной последовательности отказов) ничего не дают при тестировании ($\mathfrak{R}$-замкнутость модели), и можно ограничиться только такими тестами, в трассах которых нет повторных отказов. В общем случае «лишним» является любой отказ $\mathtt{P}$, который в тестовой трассе расположен после некоторой последовательности отказов ρ, а все действия $\mathtt{z}\in\mathtt{P}$ запрещаются отказами из ρ. Иными словами, после трассы $\mu\cdot\rho$ нажатие кнопки “P” «лишнее», поскольку реализация согласована и, для безопасной трассы, возможен единственный ответ – отказ

$P \in \mathfrak{R}$. Оптимизация заключается в замене всех трасс теста вида $\mu \cdot \rho \cdot \langle P \rangle \cdot \lambda$, где $P \in \mathfrak{R}$, $\rho \in \mathfrak{R}^*$ и $P \subseteq \cup_\circ \boldsymbol{Im}(\rho)$, на трассы $\mu \cdot \rho \cdot \lambda$.

Заметим, что такую оптимизацию можно делать и для управляемых тестов, в частности, примитивных тестов. Если кнопочные множества не пересекаются, удаление лишних отказов оставляет только согласованные трассы в тесте.

3. Неопределённость в возможном вердикте.

Можно также потребовать от оператора, 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 $\mu \cdot \lambda_f$. Если это не так, то есть 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.

Заметим, что в строгом тесте, в частности, в примитивном тесте, любая немаксимальная трасса μ, должна иметь продолжение до ***pass***-трассы. Действительно, если трасса μ немаксимальна, то она безопасна в спецификации и должна существовать кнопка “P”, безопасная в спецификации после трассы μ и удовлетворяющая условию отсутствия тупика в тесте. Если $P \in \mathfrak{R}$, то, по $\mathfrak{R}$-конвергентности спецификации, трасса μ (которая должна быть безопасной в спецификации) продолжается в спецификации либо действием $u \in P$, либо отказом $u = P$. Если же $P \in \mathfrak{Q}$, то трасса μ должна продолжаться в спецификации действием $u \in P$. В любом случае такое продолжение $\mu \cdot \langle u \rangle$ безопасно в спецификации и должно быть трассой теста (по условию отсутствия тупика). Следовательно, это максимальная трасса теста с вердиктом ***pass*** (поскольку тест строгий).

Поэтому, если все продолжения немаксимальной трассы μ строгого теста имеют один вердикт, то таким вердиктом может быть только ***pass***.

В этом последнем случае вместо оптимизации тестов, генерируемых по исходной спецификации, мы могли бы модифицировать саму спецификацию: любое продолжение $\mu\cdot\langle z\rangle\cdot\lambda$, где z безопасное после μ внешнее действие, заменить на $\mu\cdot\langle z,\gamma\rangle$. Класс безопасных реализаций при этом может расшириться: перестаём проверять действие z после μ и, тем самым, допускаем в реализации трассу $\mu\cdot\langle z,\gamma\rangle$, которую раньше запрещали. При этом все добавленные реализации будут конформны модифицированной спецификации. Это различие между исходной и модифицированной спецификациями, тем не менее, не проверяемо при тестировании, поскольку относится только к предусловию тестирования – гипотезе о безопасности.

Если рассматривать спецификацию как «руководство» по созданию конформных реализаций, то такая модификация вредна, поскольку она разрешает разрушение после трассы $\mu\cdot\langle z\rangle$, которое мы хотели бы запретить. Однако, с точки зрения генерации тестов эта модификация полезна: она оптимизирует тестирование, удаляя лишние проверки в случае, когда при любом продолжении тестирования возможен только один вердикт ***pass***.

4. Удаление «лишних» тестов из набора строгих тестов.

По Теорема 7:, для полноты тестирования достаточно, чтобы набор строгих тестов $\mathbf{TT}$ покрывал все тестовые трассы спецификации. Поэтому, если каждая трасса некоторого теста $\mathbf{T}\in\mathbf{TT}$ принадлежит какому-нибудь другому тесту набора $\mathbf{T}\subseteq\cup(\mathbf{TT}\setminus\mathbf{T})$, то тест $\mathbf{T}$ можно удалить из набора: набор $\mathbf{TT}\setminus\{\mathbf{T}\}$ также полон.

Здесь, однако, «лёд тонок»: такую оптимизацию мы можем никогда не довести до конца. Требование отсутствия «лишних» тестов в наборе может вступить в противоречие с требованием конечности времени выполнения

теста. Проблема в том, что, как правило, дерево всех тестовых трасс спецификации не является нётеровым, а тесты должны быть нётеровыми.

В качестве примера достаточно рассмотреть $\mathfrak{R}/\mathfrak{Q}$-семантику, в которой каждое внешнее действие разрешается только одной кнопкой (в частности, $\beta\gamma\delta$-семантику). Здесь тест однозначно определяется подмножеством своих трасс, безопасных в спецификации. Пусть в спецификации каждая безопасная трасса безопасно продолжается хотя бы одним действием. Тогда в любом нётеровом тесте $\mathbf{T}$, в котором есть безопасная трасса μ, нет некоторой безопасной трассы $\lambda>\mu$. Однако в полном тестовом наборе $\mathbf{TT}$ должен найтись тест $\mathbf{T}`$, в котором есть трасса λ и, следовательно, как её префикс есть трасса μ. Тем самым, $\mathbf{T}\subseteq\cup(\mathbf{TT}\setminus\{\mathbf{T}\})$.

Таким образом, можно ограничиться генерацией только редуцированных (возможно, частично) тестов. Однако, поскольку это является очевидной оптимизацией (хотя на практике важной и полезной), мы не будем на этом акцентировать внимание.

3.7.10. Алгоритмизация

Вопросы разработки практических алгоритмов генерации тестов выходят за рамки темы данной работы. Здесь нам важно лишь показать возможность алгоритмизации генерации тестов, в частности, примитивных тестов, и указать на основные идеи такой алгоритмизации.

Прежде всего, заметим, что из-за нётеровости каждого теста полный тестовый набор, как правило, бесконечен (для любой спецификации, в которой есть бесконечная трасса, в частности, порождённая маршрутом-циклом). Поэтому задачу генерации тестов полного набора естественно поставить как задачу *перечисления* этих тестов. Если для практических нужд мы имеем некоторый критерий покрытия, позволяющий отобрать конечный значимый набор тестов, то это можно сделать, фильтруя перечисляемую последовательность тестов по этому критерию.

Нам требуется алгоритмическое задание самой трассовой модели спецификации. Если трассовая модель является регулярным множеством трасс, её можно задать конечным порождающим графом. Однако это мало нам помогает в перечислении бесконечного тестового набора.

В общем случае мы будем предполагать, что спецификация Σ и протокол взаимодействия (отношение ***safe by***) задаются конечным набором *локальных алгоритмов*. Локальность алгоритма означает, что он по заданной ему трассе сообщает информацию о её продолжениях символами (внешними действиями, отказами, разрушением и дивергенцией). Достаточно, чтобы такая трасса была безопасной трассой или *трансбезопасной* трассой: пустой трассой или безопасной трассой, продолженной внешним действием. Для определения безопасности кнопки после трассы нужно уметь за конечное время определять, продолжается ли эта трасса в спецификации, во-первых, дивергенцией, и, во-вторых, разрушающими действиями (действиями и далее разрушением), принадлежащими этой кнопке. Для Ω-кнопки дополнительно нужно определять, имеется ли продолжение трассы неразрушающим действием из этой кнопки, и, если имеется, объявлена ли эта кнопка опасной или безопасной по отношению ***safe by***. Для определения безопасности после трассы продолжающего действия нужно уметь за конечное время определять наличие безопасной кнопки, которой принадлежит это действие.

В целом, прежде всего, нам нужен алгоритм, который сообщал бы, есть в спецификации трасса $\langle\gamma\rangle$ или нет. Если есть, то любое тестирование опасно и излишне, поскольку любая реализация конформна. Если нет, то в спецификации есть безопасные трассы: по крайней мере, пустая трасса безопасна. Локальные алгоритмы могут задаваться различными способами, но, в конечном счёте, они должны определять три базовых алгоритма, получающих в качестве одного из аргументов безопасную трассу $\sigma \in \boldsymbol{SafeBy}(\Sigma)$:

1) *Итератор кнопок* – алгоритм, перечисляющий кнопки, безопасные после σ. Предполагается, что множество таких кнопок *перечислимо*.

2) *Итератор продолжений* – алгоритм, перечисляющий для кнопки “P”, безопасной после σ, подмножество спецификационных продолжений $\boldsymbol{obs}(\sigma, P, \Sigma)$. Предполагается, что это подмножество *перечислимо*.
3) *Оракул* – алгоритм, разрешающий для кнопки “P”, безопасной после σ, подмножество спецификационных продолжений $\boldsymbol{obs}(\sigma, P, \Sigma)$ относительно надмножества всех потенциально возможных продолжений $\{u \mid u \in P \vee u = P \in \mathfrak{R}\}$. Предполагается, что подмножество *разрешимо* относительно надмножества (как следствие, оно *перечислимо*).

Например, пусть для спецификации Σ алфавит L конечен, а отношение ***safe by*** объявляет безопасной после трассы $\sigma \in \Sigma$ каждую $\mathfrak{Q}$-кнопку “P”, разрешающую внешнее действие z, продолжающее трассу σ в спецификации, $\sigma \cdot \langle z \rangle \in \Sigma$, и каждую $\mathfrak{R}$-кнопку “P” при дополнительном условии в обоих случаях: трасса σ не продолжается дивергенцией $\sigma \cdot \langle \Delta \rangle \notin \Sigma$, а кнопка “P” не разрушающая после трассы: $\forall t \in P \ \sigma \cdot \langle t, \gamma \rangle \notin \Sigma$. В частности, если кнопочные множества не пересекаются, отношение ***safe by*** будет только таким. Из конечности алфавита следуют конечность множества кнопок и каждого кнопочного множества. Тогда нам достаточно иметь алгоритм, который по каждой трассе $\sigma \in \Sigma$ перечисляет все базовые действия и $\mathfrak{R}$-отказы, продолжающие эту трассу, то есть перечисляет конечное множество всех продолжений $\{u \in L_{\mathfrak{R}\Delta\gamma} \mid \sigma \cdot \langle u \rangle \in \Sigma\}$. Для построения итератора кнопок достаточно перебирать все кнопки и для каждой кнопки “P” проверять, продолжается ли трасса σ дивергенцией и, если не продолжается, продолжается ли она действиями, разрешаемыми кнопкой “P”, и, если продолжается действием $z \in P$, то есть $\sigma \cdot \langle z \rangle \in \Sigma$, то является ли оно разрушающим. $\mathfrak{R}$-кнопка безопасна, если она не разрушающая, и трасса не

продолжается дивергенцией, а $\mathfrak{Q}$-кнопка, кроме этого, должна разрешать хотя бы одно продолжающее действие. Итератор продолжений строится как перечисление пересечения двух конечных множеств: множества всех продолжений и множества $\{u|u\in P \vee u=P\in\mathfrak{R}\}$. Оракул разрешает это конечное пересечение относительно конечного множества $\{u|u\in P \vee u=P\in\mathfrak{R}\}$.

В общем случае примитивный тест задаётся конечной тестовой историей σ (чередующейся последовательностью кнопок и наблюдений) и оракулом, который по каждой кнопке "$\sigma(i)$" и предшествующему ей префиксу трассы $\sigma[1..i-1]{\downarrow}L_{\mathfrak{R}}$ определяет правильность полученного после нажатия этой кнопки наблюдения u. Если наблюдение u неправильно, выдаётся вердикт ***fail***. Если наблюдение u правильно и оно последнее в тестовой истории $u=\sigma(i+1)$ & $i+1=|\sigma|$ или ответвляется от тестовой истории $u\neq\sigma(i+1)$, то выдаётся вердикт ***pass***. Если наблюдение u правильно, продолжает тестовую историю, но не последнее в ней, $u=\sigma(i+1)$ & $i+1<|\sigma|$, то нажимается следующая кнопка "$\sigma(i+2)$".

Таким образом, для перечисления полного тестового набора всех примитивных тестов нам достаточно перечислить все тестовые истории, подтрассы которых безопасны. Это выполняется как «диагональный» процесс, чередующий вызов итераторов кнопок и продолжений.

Для каждой тестовой истории μ итератор кнопок задаёт нумерацию кнопок, безопасных после трассы $\mu{\downarrow}L_{\mathfrak{R}}$. Итератор продолжений, в свою очередь, задаёт после этой трассы и некоторой безопасной кнопки "P" нумерацию продолжающих наблюдений (внешних действий и $\mathfrak{R}$-отказов). Для тестовой истории σ её индексом назовём сумму номеров её кнопок и продолжающих наблюдений: номер i-ого символа $\sigma(i)$ определяется

непосредственно предшествующим ему префиксом трассы $\sigma[1..i-1]{\downarrow}L_{\mathfrak{R}}$, если $\sigma(i)$ это кнопка, или дополнительно кнопкой $\sigma(i-1)$, если $\sigma(i)$ это наблюдение. Очевидно, что множество тестовых историй с данным индексом конечно, и его можно перебрать алгоритмически, используя итераторы кнопок и продолжений. Тогда итерация тестовых историй реализуется двумя вложенными циклами: во внешнем цикле перечисляем индекс, а во внутреннем – истории с этим индексом; индекс 0 имеет пустая история.

3.8. Выводы

В этой главе показано, что трассовая теория достаточна для формулировки отношения безопасности и безопасной конформности. Были определены безопасность в реализации (отношение ***safe in***), безопасность в спецификации для того или иного протокола взаимодействия (отношение ***safe by***), гипотеза о безопасности (отношение ***safe for***), определяющая класс безопасных реализаций, и безопасная конформность ***saco***.

Для конформности ***saco*** определено понятие трассового теста и выяснены условия конечности времени выполнения теста. Также введено понятие управляемого (без лишнего недетерминизма) теста. Далее на трассовом уровне определено отношение «реализация проходит тест», значимые, исчерпывающие и полные наборы тестов. Выделен подкласс строгих тестов и доказано утверждение о полном наборе строгих управляемых тестов. Для примитивных тестов этот набор содержит только ограниченные управляемые строгие тесты. После этого рассмотрены вопросы оптимизации тестов и набора тестов, которые могут использоваться на практике. Глава завершается кратким рассмотрением вопроса об алгоритмическом задании трассовой модели и алгоритмической генерации полного тестового набора.

Новыми являются следующие результаты:

1) Определение модели наблюдаемых трасс для заданной семантики взаимодействия, которые состоят из действий и отказов и могут

заканчиваться дивергенцией или разрушением. Даже без учёта дивергенции и разрушения, такое определение впервые даётся интенсионально, а не генетически (через LTS), что позволяет строить теорию трассовых моделей безотносительно к теории LTS-моделей.

2) Утверждение о разложении трассовой модели на стабильные и контрстабильные деревья.
3) Понятие безопасного тестирования как тестирования, избегающего разрушения и не продолжающего тестовый эксперимент в том случае, когда возможна дивергенция. Это является хорошей альтернативой полному запрету на дивергенцию и разрушение, что позволяет расширить домен конформности.
4) Введение в научный оборот гипотезы о безопасности – такой гипотезы о реализации, которая позволяет безопасно тестировать её для заданной спецификации.
5) Понятие безопасной конформности ***saco*** (*safe conformance*), которая позволяет декларировать в спецификации дивергенцию и разрушение, опирается на гипотезу о безопасности и параметризуется семантикой взаимодействия. Частные конформности, основанные на семантиках наблюдаемого поведения (без учёта состояний), такие как отношение ***ioco***, можно рассматривать как частные случаи отношения ***saco***. Допущение дивергенции позволяет решить проблему, характерную для отношений типа ***ioco***, которая заключается в выходе за пределы домена конформности при композиции (появление дивергенции как результата бесконечного взаимодействия компонентов).
6) Метод генерации полного набора тестов для безопасной конформности по заданной спецификации, являющийся обобщением аналогичных методов для частных конформностей (в том числе, отношения ***ioco***). Метод алгоритмизуем при достаточно слабых ограничениях на спецификацию. Отличие от аналогичных методов, во-первых, в том, что спецификация может быть задана в виде модели наблюдаемых трасс, а не в виде LTS, и, во-

вторых, в том, что обеспечивается безопасное тестирование даже, если в реализации есть дивергенция и разрушение, при условии, что реализация удовлетворяет гипотезе о безопасности.

Глава 4. Система помеченных переходов (LTS-модель)

Структура главы:

1. Определение LTS
2. $\mathfrak{R}$-трассы и *F*-трассы LTS-модели
3. Объединение множества LTS-моделей
4. Машина тестирования и LTS-модель
5. Эквивалентность трассовой и LTS-моделей
6. Безопасность и конформность
7. Композиция LTS и тесты
8. Выводы

В этой главе в качестве модели рассматривается *система помеченных переходов* – LTS (*Labelled Transition System*), состоящая из состояний и переходов между состояниями, помеченных символами внешних действий или символом τ для внутренних действий. Отказ P возможен в состоянии, из которого не выходят τ-переходы и переходы по внешним действиям $z \in P$.

Развивая эту модель, мы также разрешаем в LTS разрушение – переход по символу γ. Теперь в стабильном состоянии не должно быть также γ-переходов. Дивергенция как бесконечная цепочка τ-переходов сохраняется в LTS и моделируется Δ-действием при построении $\mathfrak{R}$-трасс LTS.

LTS более естественно «укладывается» в чёрный ящик машины тестирования, чем трассовая модель. Тем не менее, мы покажем, что множество $\mathfrak{R}$-трасс LTS является трассовой $\mathfrak{R}$-моделью и, соответственно, $\mathfrak{R}$-проекцией ***F***-модели как множества ***F***-трасс LTS-модели. Также докажем обратное утверждение: любая ***F***-модель является множеством ***F***-трасс некоторой LTS. Соответственно, $\mathfrak{R}$-проекция ***F***-модели совпадает с множеством $\mathfrak{R}$-трасс LTS,

имеющей то же множество ***F***-трасс, а $\mathfrak{R}$-трасса в этой LTS продолжается ненаблюдаемым $\mathfrak{Q}$-отказом тогда и только тогда, когда она продолжается этим $\mathfrak{Q}$-отказом в исходной ***F***-модели. Тем самым будет показана эквивалентность трассовой и LTS-моделей как эквивалентность интенсионального и генетического (через LTS) определений модели наблюдаемых трасс.

Опираясь на такую эквивалентность трассовой и LTS-моделей, мы определим гипотезу о безопасности, конформность и генерацию тестов для LTS, заимствуя соответствующие определения из трассовой теории.

LTS-теория имеет то преимущество перед теорией $\mathfrak{R}$-трасс, что в ней можно определить композицию составной модели из моделей-компонентов, используя оператор параллельной композиции алгебры процессов (CCS в данной работе). Соответственно, взаимодействие в LTS-теории выглядит как композиция LTS-реализации и LTS-окружения, а тестирование – как композиция LTS-реализации и LTS-теста. Мы докажем утверждение об эквивалентности отношений «проходит» в трассовой и LTS-теориях. С этой точки зрения мы проинтерпретируем $\mathfrak{R}$/$\mathfrak{Q}$-машину тестирования как ограничение на LTS-окружение/тест.

4.1. Определение LTS-модели

Определение 34:

- Будем считать, что символ $\tau \notin Z_\gamma$, и будем называть его символом внутреннего действия.
- Для произвольного множества A будем обозначать: $A_\tau = A \cup \{\tau\}$.
- *Системой помеченных переходов* или LTS (*Labelled Transition System*) будем называть совокупность $\mathbf{S} = LTS(V_\mathbf{S}, L, E_\mathbf{S}, s_0)$, где:

  $V_\mathbf{S}$ – непустое множество *состояний*,

  L – множество *символов* (алфавит символов),

$\tau \notin L$,

$E_{\mathbf{S}} \subseteq V_{\mathbf{S}} \times L_\tau \times V_{\mathbf{S}}$ – множество переходов,

$s_0 \in V_{\mathbf{S}}$ – начальное состояние.

- Класс LTS с заданным алфавитом L обозначим ***LTS***(L).
- Класс всех LTS обозначим ***LTS***.
- В рамках данной теории символ τ будем считать *праэлементом* (объектом, не являющимся классом).

□

Классы ***LTS*** и ***LTS***(L) не является множествами, так как не являются множествами класс всех состояний всех LTS и класс всех состояний всех LTS в алфавите L. Это связано с тем, что мы не налагаем никаких ограничений на природу состояний отдельной LTS. Если бы класс всех LTS считался множеством, это привело бы к тем или иным вариантам известных парадоксов наивной теории множеств. Один из таких вариантов рассматривается ниже в разделе 5.4.[12]

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: $V_{\mathbf{S}}, V_{\mathbf{T}}, \ldots$ и $E_{\mathbf{S}}, E_{\mathbf{T}}, \ldots$, а их начальные состояния соответствующими строчными буквами с нижним индексом "$_0$": $s_0, t_0, \ldots$. Для того, чтобы формальная запись была более компактной, мы будем использовать сокращения и писать ***R*** вместо ***runs*** (маршруты), ***T*** вместо ***traces*** (трассы внешних действий) и ***F*** вместо ***Ftraces*** (трассы с отказами).

---

[12] С той же целью обхода этих парадоксов иногда вместо «система» говорят «пространство» (labeled transition space в [90]), а для задания отдельной LTS указывают одно из состояний множества состояний этого LTS-пространства в качестве начального состояния указываемой LTS. Мы выбрали вариант с NBG теорией множеств для того, чтобы не быть ограниченными в различных преобразованиях LTS, которые по одной или нескольким LTS строят новую LTS. В частности, если строится LTS, состояния которой – это все подмножества состояний исходной LTS, то возрастает мощность множества состояний LTS, и, в конце концов, она перестаёт «помещаться» в LTS-пространстве.

Определение 35: Для $\mathbf{S}=LTS(V_{\mathbf{S}}, L, E_{\mathbf{S}}, s_0)$, $s, s` \in V_{\mathbf{S}}$, $U \subseteq V_{\mathbf{S}}$, $t \in L_\tau$, $\sigma \in L^\omega$ определим:

- ***pre***$(s,t,s`)$ $=_{def}$ s – пресостояние перехода.
- ***post***$(s,t,s`)$ $=_{def}$ $s`$ – постсостояние перехода.
- ***symbol***$(s,t,s`)$ $=_{def}$ t – символ перехода.
- $s \text{—} t \rightarrow s` =_{def} (s,t,s`) \in E_{\mathbf{S}}$.[13]
- $s \text{—} t \rightarrow =_{def} \exists s` \; s \text{—} t \rightarrow s`$.
- $s \text{—} t \nrightarrow s` =_{def} \neg \; s \text{—} t \rightarrow s`$.
- $s \text{—} t \nrightarrow =_{def} \neg \; s \text{—} t \rightarrow$.
- $R \in E_{\mathbf{S}}^{\omega}$ будем называть маршрутом с началом в s, если $R=\epsilon$ или

  $R \neq \epsilon \;\&\; \textbf{\textit{pre}} \circ R(1)=s \;\&\; \forall i \in [2..|R|] \; \textbf{\textit{post}} \circ R(i-1)=\textbf{\textit{pre}} \circ R(i)$.

  - Начальное состояние подразумевается для любого маршрута, в том числе пустого. Если нужно указать начальное состояние s пустого маршрута, будем обозначать этот пустой маршрут как (s,ϵ).
  - Концом конечного маршрута $R \in E_{\mathbf{S}}^{*}$ назовём постсостояние его последнего перехода, если маршрут не пуст, или его начало, если маршрут пуст; конец маршрута обозначим $\omega(R)$:

    $R \neq \epsilon$: $\omega(R) =_{def} \textbf{\textit{post}} \circ R(|R|)$, $\omega((s,\epsilon)) =_{def} s$.
- $\boldsymbol{R}^{\omega}(s)$ – множество маршрутов с началом в s.
- $\boldsymbol{R}^{\infty}(s)$ – множество бесконечных маршрутов с началом в s.
- $\boldsymbol{R}(s)$ – множество конечных маршрутов с началом в s.[14]
- $\boldsymbol{R}^{\omega}(\mathbf{S}) =_{def} \boldsymbol{R}^{\omega}(s_0)$.

---

[13] Для выразительности записи там, где это не приведёт к недоразумениям, мы, допуская вольность речи, будем писать $s \text{—} z \rightarrow s`$ вместо $(s,z,s`)$, а не вместо $(s,t,s`) \in E_{\mathbf{S}}$.

[14] Очевидно, $\boldsymbol{R}^{\omega}(s)$ и $\boldsymbol{R}(s)$ деревья, а $\boldsymbol{R}^{\infty}(s)$ не дерево.

- $\boldsymbol{R}^{\infty}(\mathbf{s}) =_{\text{def}} \boldsymbol{R}^{\infty}(s_0)$.
- $\boldsymbol{R}(\mathbf{s}) =_{\text{def}} \boldsymbol{R}(s_0)$.
- Для $R \in \boldsymbol{R}^{\omega}(s)$:
  - $\boldsymbol{T}(R) =_{\text{def}} (\boldsymbol{symbol} \circ R) \uparrow \{\tau\}$ – трасса маршрута;
  - R τ-маршрут $=_{\text{def}} |\boldsymbol{T}(R)| = \epsilon$.
- $s = \sigma \Rightarrow s` =_{\text{def}} \exists R \in \boldsymbol{R}(s)\ \ \sigma = \boldsymbol{T}(R)\ \ \&\ \ s` = \omega(R)$.
- $s = \sigma \Rightarrow =_{\text{def}} \exists s`\ \ s = \sigma \Rightarrow s`$.
- $s = \sigma \not\Rightarrow s` =_{\text{def}} \neg\ s = \sigma \Rightarrow s`$.
- $s = \sigma \not\Rightarrow =_{\text{def}} \neg\ s = \sigma \Rightarrow$.
- $s = \Rightarrow s` =_{\text{def}} s = \epsilon \Rightarrow s`$.
- $s = \Rightarrow =_{\text{def}} s = \epsilon \Rightarrow$.
- $s = \not\Rightarrow =_{\text{def}} s = \epsilon \not\Rightarrow$.
- s ***after*** $\sigma =_{\text{def}} \{s` \in V_{\mathbf{s}} \mid s = \sigma \Rightarrow s`\}$.
- $\mathbf{s}$ ***after*** $\sigma =_{\text{def}} s_0$ ***after*** σ.[15]
- $\boldsymbol{der}(s) =_{\text{def}} \cup\{s$ ***after*** $\sigma \mid \sigma \in L^{*}\}$ – состояния, достижимые из s.
- $\boldsymbol{der}(\mathbf{s}) =_{\text{def}} \boldsymbol{der}(s_0)$ – достижимые состояния $\mathbf{s}$.
- $\boldsymbol{init}(s) =_{\text{def}} \{t \in L_{\tau} \mid s \text{—}t\rightarrow\}$.
- $\boldsymbol{T}^{\omega}(s) =_{\text{def}} \{\boldsymbol{T}(R) \mid R \in \boldsymbol{R}^{\omega}(s)\}$.
- $\boldsymbol{T}^{\infty}(s) =_{\text{def}} \{\boldsymbol{T}(R) \mid R \in \boldsymbol{R}^{\infty}(s)\}$.
- $\boldsymbol{T}(s) =_{\text{def}} \{\boldsymbol{T}(R) \mid R \in \boldsymbol{R}(s)\}$.[16]
- $\boldsymbol{T}^{\omega}(\mathbf{s}) =_{\text{def}} \boldsymbol{T}^{\omega}(s_0)$ – множество трасс $\mathbf{s}$.

---

[15] Оператор ***after*** для LTS не входит в противоречие с оператором ***after*** для деревьев последовательностей, поскольку LTS не является деревом последовательностей.

[16] Очевидно, $\boldsymbol{T}(s) \in \boldsymbol{Trees}(L)$, $\boldsymbol{T}^{\omega}(s) \in \boldsymbol{Trees}^{\omega}(L)$, а $\boldsymbol{T}^{\infty}(s)$ не дерево.

- $\boldsymbol{T}^{\infty}(\mathbf{S}) =_{\text{def}} \boldsymbol{T}^{\infty}(s_0)$ – множество бесконечных трасс $\mathbf{S}$.
- $\boldsymbol{T}(\mathbf{S}) =_{\text{def}} \boldsymbol{T}(s_0)$ – множество конечных трасс $\mathbf{S}$.
- s *терминально* $=_{\text{def}} \boldsymbol{init}(s)=\varnothing$.
- *Конвергентность*: $s\downarrow =_{\text{def}} \forall R\in \boldsymbol{R}^{\infty}(s)\ \boldsymbol{T}(R)\neq\epsilon$

  – в состоянии s не начинается бесконечный τ-маршрут;

  $$U\downarrow =_{\text{def}} \forall s\in U\ s\downarrow.$$
- *Дивергентность*: $s\uparrow =_{\text{def}} \neg(s\downarrow)$;

  $$U\uparrow =_{\text{def}} \exists s\in U\ s\uparrow.$$
- *Строгая конвергентность*: $\mathbf{S}\Downarrow =_{\text{def}} \boldsymbol{der}(\mathbf{S})\downarrow$.
- s *детерминировано* $=_{\text{def}} \forall\sigma\in L^{*}\ |s\ \boldsymbol{after}\ \sigma|\leq 1$.
- $\mathbf{S}$ *детерминирована* $=_{\text{def}}$ s_0 детерминировано.
- $\mathbf{S}$ *конечна* $=_{\text{def}} |\{e\in E_{\mathbf{S}}\,|\,\boldsymbol{pre}(e)\in\boldsymbol{der}(\mathbf{S})\}|\neq\infty$

  – конечно множество переходов из достижимых состояний.
- $\mathbf{S}$ *является LTS-деревом* $=_{\text{def}} \forall s\in\boldsymbol{der}(\mathbf{S})\ \exists!R\in\boldsymbol{R}(\mathbf{S})\ \omega(R)=s$

  – в каждом достижимом состоянии заканчивается только один маршрут, начинающийся в начальном состоянии (в начальном состоянии заканчивается только пустой маршрут).

□

Определение 36: Две LTS в одном алфавите $\mathbf{A},\mathbf{B}\in\boldsymbol{LTS}(L)$ называют *изоморфными*, если существует такая биекция множеств достижимых состояний $\boldsymbol{f}:\boldsymbol{der}(\mathbf{A})\to\boldsymbol{der}(\mathbf{B})$, что $\boldsymbol{f}(a_0)=b_0$ и

$\forall a,a`\in\boldsymbol{der}(\mathbf{A})\ \forall t\in L_{\tau}\ (a\text{—}t\to a` \Leftrightarrow \boldsymbol{f}(a)\text{—}t\to\boldsymbol{f}(a`))$.

Биекцию $\boldsymbol{f}$ называют в этом случае *изоморфизмом*.

□

Как правило, мы будем рассматривать LTS с точностью до изоморфизма.

Определение 37: Будем говорить, что LTS $\mathbf{A}=LTS(V_{\mathbf{A}},A,E_{\mathbf{A}},a_0)$ является под-LTS LTS $\mathbf{B}=LTS(V_{\mathbf{B}},B,E_{\mathbf{B}},b_0)$, порождённой множеством переходов $E_{\mathbf{A}}$, если $A=B$, $E_{\mathbf{A}}\subseteq E_{\mathbf{B}}$ и $a_0=b_0$.[17]

□

Определение 38: LTS-моделью будем называть LTS в алфавите L_γ, где $L\subseteq Z$ алфавит внешних действий. Класс всех LTS-моделей в алфавите L обозначим $\boldsymbol{LTS}(L_\gamma)$. Класс всех LTS-моделей (во всех алфавитах) обозначим $\boldsymbol{LTS}_\gamma$.

□

Определение 39: Для LTS-модели $\mathbf{S}$ и её состояния $s\in V_{\mathbf{S}}$ определим:

· s *стабильно* $=_{def}\ s\text{—}\tau\gamma\not\rightarrow =_{def}\ s\text{—}\tau\not\rightarrow\ \&\ s\text{—}\gamma\not\rightarrow$.

· s *нестабильно* $=_{def}\ s\text{—}\tau\gamma\rightarrow =_{def}\ s\text{—}\tau\rightarrow\ \vee\ s\text{—}\gamma\rightarrow$.

□

Можно отметить, что автомат Мили (Mealy), часто используемый в качестве модели тестируемой реактивной системы, также можно трактовать как разновидность LTS-модели. В таком автомате следующий стимул можно давать только после получения реакции на предыдущий стимул, и после выдачи одной реакции не выдаётся другая (или та же самая) реакция до получения следующего стимула. Автомат Мили можно понимать как LTS, в которой переход помечается парой (стимул, реакция) и нет τ-переходов. Учитывая, что стимулы и реакции – это различные внешние действия, соответствующую LTS-модель можно построить, добавив новые промежуточные состояния: для каждого состояния s и стимула $?x$ добавляется состояние $(s,?x)$. Тогда каждый переход $s\text{—}(?x,!y)\rightarrow s`$ преобразуется в пару переходов $s\text{—}?x\rightarrow(s,?x)\text{—}!y\rightarrow s`$. Во всех старых состояниях есть переходы только по стимулам, а во всех новых состояниях есть переходы только по реакциям.

[17] Разумеется, предполагается, что все пре- и пост-состояния переходов из $E_{\mathbf{A}}$ принадлежат $V_{\mathbf{A}}$, тем самым, $V_{\mathbf{A}}\subseteq V_{\mathbf{B}}$.

Соответствующие отказы: стационарность в старых состояниях, соответствующая правилу «после реакции только стимул», и блокировки всех стимулов в новых состояниях, соответствующие правилу «после стимула только реакция». Начальное состояние остаётся старым, что соответствует правилу «сначала стимул, а реакция – в ответ».

В случае реактивных систем, когда внешние действия разделяются на стимулы и реакции, иногда говорят об *Input-Output Labelled Transition System* (IOLTS) [105]. В литературе можно встретить и другие названия: *Input-Output Transition System* (IOTS) [149], *Input-Output Automaton* (IOA) [117], *Input-Output State Machines* (IOSM) [136] и т.п. Различия между этими понятиями незначительны.

4.2. $\mathfrak{R}$-трассы и *F*-трассы LTS-модели

Структура раздела:

1. Преобразование LTS-модели в трассовую модель
2. Распространение LTS-обозначений на $\mathfrak{R}$-трассы
3. $\mathfrak{R}$-маршруты

В этом разделе мы определим ***F***-трассы LTS-модели с помощью преобразования $\boldsymbol{LTS}(\mathtt{L}_{\gamma}) \rightarrow \boldsymbol{LTS}(\mathtt{L}_{\mathcal{P}(\mathtt{L})\Delta\gamma})$ и взятия трасс преобразованной LTS. Мы покажем, что множество ***F***-трасс LTS-модели является ***F***-моделью (Рис.36). Соответственно, $\mathfrak{R}$-трассы LTS-модели как $\mathfrak{R}$-проекция её ***F***-трасс является $\mathfrak{R}$-моделью.

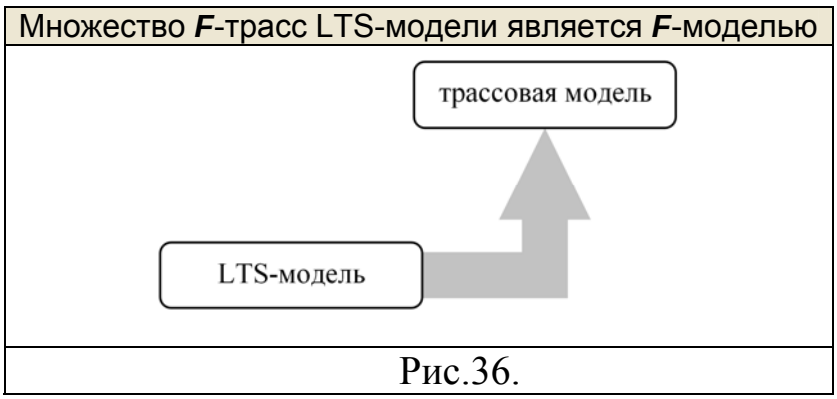


Рис.36.

4.2.1. Преобразование LTS-модели в трассовую модель

Нестабильное состояние LTS не порождает отказов, а в стабильном состоянии порождается отказ P, если в этом состоянии не определены переходы по внешним действиям $z \in P$.

Определение 40: Для $\mathbf{s} \in \boldsymbol{LTS}(L_\gamma)$ и состояния $s \in V_{\mathbf{s}}$ обозначим множество $\boldsymbol{F}$-отказов, порождаемых состоянием s:

$$s \xrightarrow{\tau\gamma} : \boldsymbol{f}(s) =_{def} \varnothing,$$

$$s \xrightarrow{\tau\gamma} : \boldsymbol{f}(s) =_{def} \mathcal{P}(L \setminus \boldsymbol{init}(s)).$$

□

При преобразовании отказы изображаются в LTS добавлением в каждом стабильном состоянии переходов-петель по каждому порождаемому этим состоянием отказу. Для того, чтобы разрушение было только последним символом $\boldsymbol{F}$-трасс, перенаправляем все γ-переходы в добавленное терминальное состояние $\mathbf{t}$. Моделируя дивергенцию Δ-действием, в каждом дивергентном состоянии добавляем переход по Δ-действию в $\mathbf{t}$-состояние.

Определение 41: Для алфавита L определим преобразование $\boldsymbol{L2L_F}: \boldsymbol{LTS}(L_\gamma) \to \boldsymbol{LTS}(L_{\mathcal{P}(L)\Delta\gamma})$. Для $\mathbf{S}=LTS(V_{\mathbf{S}}, L_\gamma, E_{\mathbf{S}}, s_0)$ $\boldsymbol{L2L_F}(\mathbf{S})=\mathbf{M}=LTS(V_{\mathbf{S}}\cup\{\mathbf{t}\}, L_{\mathcal{P}(L)\Delta\gamma}, E_{\mathbf{M}}, s_0)$, где $\mathbf{t}\notin V_{\mathbf{S}}$,[18] а $E_{\mathbf{M}}$ – наименьшее множество, порождаемое следующими правилами вывода: $\forall s,s`\in V_{\mathbf{S}}\ \forall z\ \forall P$

$z\in L_\tau\ \&\ s\text{—}z\to s` \vdash s\text{—}z\to s`$,

$P\in \boldsymbol{f}(s) \vdash s\text{—}P\to s$,

$s\text{—}\gamma\to \vdash s\text{—}\gamma\to\mathbf{t}$,

$s\uparrow \vdash s\text{—}\Delta\to\mathbf{t}$.

□

Определение 42: Пусть задана $\mathbf{S}\in\boldsymbol{LTS}(L_\gamma)$.

- $\boldsymbol{F}$-трассой LTS $\mathbf{S}$ будем называть трассу LTS $\boldsymbol{L2L_F}(\mathbf{S})$:
  для $s\in V_{\mathbf{S}}$: $\boldsymbol{F}(s) =_{\text{def}} \boldsymbol{T}(s)$ в $\boldsymbol{L2L_F}(\mathbf{S})$,
  $\boldsymbol{F}(\mathbf{S}) =_{\text{def}} \boldsymbol{F}(s_0)$.
- Для $\mathfrak{R}\subseteq\mathcal{P}(L)$ будем называть $\mathfrak{R}$-трассой LTS её $\boldsymbol{F}$-трассу, содержащую отказы только из семейства $\mathfrak{R}$.

□

Множество $\mathfrak{R}$-трасс LTS можно определить также как множество трасс другого преобразования $\boldsymbol{L2L_{\mathfrak{R}}}(\mathbf{S})$, которое отличается от преобразования $\boldsymbol{L2L_F}(\mathbf{S})$ только тем, что в стабильных состояниях проводятся петли не по всем порождаемым этим состоянием отказам, а только по тем, которые принадлежат семейству $\mathfrak{R}$.

---

[18] Если $\mathbf{t}\in V_{\mathbf{S}}$, то можно переименовать это состояние, что эквивалентно рассмотрению изоморфной LTS.

Покажем, что множество ***F***-трасс LTS является ***F***-моделью. Соответственно, множество её $\mathfrak{R}$-трасс будет $\mathfrak{R}$-моделью.

Теорема 9: Множество ***F***-трасс LTS-модели является ***F***-моделью.

□455

4.2.2. Распространение LTS-обозначений на $\mathfrak{R}$-трассы

Распространим LTS-обозначения, использующие трассы, на ***F***- и $\mathfrak{R}$-трассы. Такими обозначениями являются: $s=\sigma\Rightarrow s`$, а также производные от этого обозначения $s=\sigma\Rightarrow$, $s=\sigma\not\Rightarrow s`$, $s=\sigma\not\Rightarrow$, s ***after*** σ и $\mathbf{S}$ ***after*** σ. Обозначение $s=\sigma\Rightarrow s`$ имеет один и тот же смысл для LTS $\mathbf{S}$ и $\boldsymbol{L2L_F}(\mathbf{S})$, если ***F***-трасса σ содержит только внешние действия. Если σ содержит отказы или заканчивается Δ-символом, двойная стрелка, очевидно, применяется к LTS $\boldsymbol{L2L_F}(\mathbf{S})$, поскольку в LTS $\mathbf{S}$ нет переходов по Δ-символу и отказам. Двусмысленность возможна лишь в том случае, когда σ не содержит отказов и символа Δ и заканчивается разрушением γ, что происходит из-за того, что все γ-переходы в LTS $\boldsymbol{L2L_F}(\mathbf{S})$ перенаправляются в $\mathbf{t}$-состояние. Мы будем считать, что в этом случае имеется в виду LTS $\boldsymbol{L2L_F}(\mathbf{S})$. Тем самым, мы переопределяем двойную стрелку $s=\sigma\Rightarrow s`$ и производные обозначения для LTS $\mathbf{S}$ и трасс без отказов и Δ-символа, заканчивающихся разрушением.

Определение 43: Для $\mathbf{S}\in\boldsymbol{LTS}(L_\gamma)$, $s\in V_{\mathbf{S}}$, $s`\in V_{\mathbf{S}}\cup\{\mathbf{t}\}$, $\sigma\in L_{\mathcal{P}(L)\Delta\gamma}{}^{\omega}$ обозначим: $s=\sigma\Rightarrow s` =_{def} s=\sigma\Rightarrow s`$ в LTS $\boldsymbol{L2L_F}(\mathbf{S})$ по Определению 35.

□

Одинарные стрелки $s\text{—}t\rightarrow s`$, $s\text{—}t\rightarrow$, $s\text{—}t\not\rightarrow s`$ и $s\text{—}t\not\rightarrow$, по-прежнему, означают наличие или отсутствие переходов и будут пониматься в

контексте исходной LTS $\mathbf{s}$. В частности, $s=\langle\gamma\rangle\Rightarrow$ влечёт s ***after*** $\langle\gamma\rangle = \{\mathbf{t}\}$, но $s\text{—}\gamma\rightarrow s'$ влечёт $s'\neq\mathbf{t}$ (так как $\mathbf{t}\notin V_{\mathbf{s}}$).

4.2.3. $\mathfrak{R}$-маршруты

Определение 44: Пусть задана $\mathbf{s}\in\boldsymbol{LTS}(L_\gamma)$. ***F***-маршрутом LTS $\mathbf{s}$ будем называть маршрут LTS $\boldsymbol{L2L_F}(\mathbf{s})$. Для $\mathfrak{R}\subseteq\mathcal{P}(L)$ будем называть $\mathfrak{R}$-маршрутом LTS $\mathbf{s}$ её ***F***-маршрут, в котором нет переходов по отказам из $\mathcal{P}(L)\setminus\mathfrak{R}$.

□

Определение 45: Пусть задана $\mathbf{s}\in\boldsymbol{LTS}(L_\gamma)$.

- Для маршрута R LTS $\mathbf{s}$ определим множество *ассоциированных* с ним ***F***-маршрутов, которые получаются из R последовательностью операций:
  1. первая обязательная операция: замена первого γ-перехода $s\text{—}\gamma\rightarrow s'\in\boldsymbol{Im}(R)$ на переход $s\text{—}\gamma\rightarrow\mathbf{t}$ и удаление всех последующих переходов маршрута R;
  2. последовательность (быть может, пустая) вставок переходов вида $s\text{—}P\rightarrow s$ в начало получившегося после предшествующих операций маршрута, если s начальное состояние маршрута, или после перехода с постсостоянием s, при условии (в обоих случаях), что s стабильное состояние, а $P\in\boldsymbol{f}(s)$;
  3. последняя необязательная операция: добавление в конец получившегося после предшествующих операций маршрута перехода $s\text{—}\Delta\rightarrow\mathbf{t}$, если этот маршрут заканчивается в дивергентном состоянии.

- Для $\mathfrak{R}\subseteq\mathcal{P}(L)$ будем говорить об ассоциированных $\mathfrak{R}$-маршрутах как ***F***-маршрутах, в которых нет переходов по отказам из $\mathcal{P}(L)\setminus\mathfrak{R}$.
- ***F***-трассой маршрута будем называть трассу ассоциированного с ним ***F***-маршрута. Множество таких ***F***-трасс будем обозначать $\boldsymbol{F}(R)$.
- $\mathfrak{R}$-трассой маршрута будем называть трассу ассоциированного с ним $\mathfrak{R}$-маршрута, то есть трассу из $\mathfrak{R}$-проекции $\boldsymbol{F}(R)^{\downarrow L_{\mathfrak{R}\Delta\gamma}}$.

□

Из этих определений вытекает очевидное следствие: множество $\mathfrak{R}$-трасс (***F***-трасс) LTS равно объединению множеств $\mathfrak{R}$-трасс (***F***-трасс) всех её маршрутов:

$\boldsymbol{F}(\mathbf{s})=\cup_{\circ}\boldsymbol{F}_{\circ}\boldsymbol{R}(\mathbf{s})$, $\boldsymbol{F}(\mathbf{s})^{\downarrow L_{\mathfrak{R}\Delta\gamma}}=\cup\{\boldsymbol{F}(R)^{\downarrow L_{\mathfrak{R}\Delta\gamma}} \mid R\in\boldsymbol{R}(\mathbf{s})\}$.

4.3. Объединение множества LTS-моделей

Объединение множества LTS можно определить с помощью введения нового начального состояния и проведения из него τ-переходов во все начальные состояния объединяемых LTS (Рис.37) .

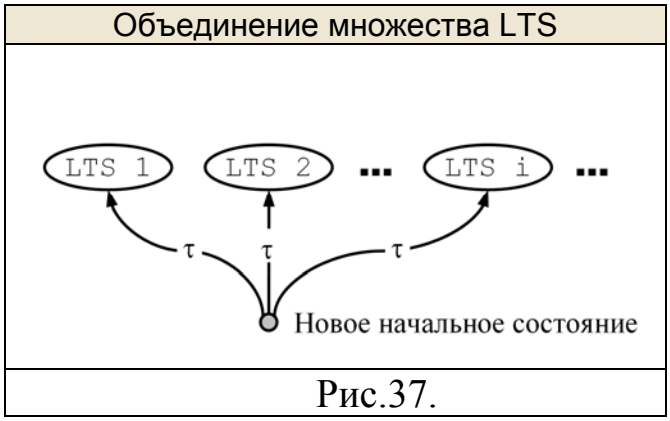


Рис.37.

Определение 46: Объединение множества LTS $\mathbf{S}_i=LTS(V_i,L_i,E_i,s_{i0})$, где $i\in I$, определим как LTS $\mathbf{S}=LTS(V,L,E,s_0)$, где

$s_0\notin\cup\{\{i\}\times V_i \mid i\in I\}$, $V=\{s_0\}\cup\cup\{\{i\}\times V_i \mid i\in I\}$,

$L = \cup\{L_i \mid i\in I\}$,

$E = \{(s_0,\tau,(i,s_{i0})) \mid i\in I\}$

$\cup\cup\{((i,s),z,(i,s`)) \mid i\in I \ \&\ (s,z,s`)\in E_i\}$.

□

Теорема 10: Объединение множества LTS-моделей $\mathbf{S}_i$ в одном алфавите L, где $i\in I$, является LTS-моделью $\mathbf{S}$ в том же алфавите L. ***F***-модель объединения множества LTS совпадает с объединением ***F***-моделей объединяемых LTS: $\boldsymbol{F}(\mathbf{S})=\cup\{\boldsymbol{F}(\mathbf{S}_i) \mid i\in I\}$.

□456

Заметим, что, если объединяются LTS в разных алфавитах, то для получения ***F***-модели объединения нужно объединять не ***F***-модели LTS-компонентов, а их расширения до ***F***-моделей в объединённом алфавите L. Это эквивалентно переходу от LTS $\mathbf{S}_i=LTS(V_i,L_i,E_i,s_{i0})$ к LTS $\mathbf{S}_i`=LTS(V_i,L,E_i,s_{i0})$, и тогда $\boldsymbol{F}(\mathbf{S})=\cup\{\boldsymbol{F}(\mathbf{S}_i`) \mid i\in I\}$.

$\mathfrak{R}$-модель объединения LTS совпадает с объединением $\mathfrak{R}$-моделей объединяемых LTS: $\boldsymbol{F}(\mathbf{S})^{\downarrow}L_{\mathfrak{R}\Delta\gamma}=\cup\{\boldsymbol{F}(\mathbf{S}_i)^{\downarrow}L_{\mathfrak{R}\Delta\gamma} \mid i\in I\})$. Если объединяются LTS в разных алфавитах, но семейство $\mathfrak{R}$ одно и то же, то $\mathfrak{R}$-модель LTS $\mathbf{S}_i$ не меняется при переходе от алфавита L_i к объединённому алфавиту L.

Заметим, что объединение собственного класса (не множества) LTS нельзя определить, поскольку класс их состояний может быть собственным (не множеством), а в LTS класс её состояний является множеством.

4.4. Машина тестирования и LTS-модель

Теперь мы покажем, что любая LTS-модель может функционировать в качестве реализации внутри машины тестирования (Рис.38). Для этого нам нужно определить правила работы такой LTS.

Рис.38.

Определение 47: Пусть в алфавите L задана $\mathfrak{R}/\mathfrak{Q}$-семантика и $\mathbf{s}\in \boldsymbol{LTS}(L_\gamma)$. Правила работы LTS $\mathbf{s}$ внутри «чёрного ящика» $\mathfrak{R}/\mathfrak{Q}$-машины тестирования следующие:

- В каждый момент времени LTS находится в некотором состоянии $s\in \boldsymbol{der}(\mathbf{s})$; в начальный момент времени LTS находится в начальном состоянии s_0.
- Если LTS находится в состоянии s и оператором машины разрешено множество внешних действий P (нажата кнопка “P” или нет нажатой кнопки и $P=\varnothing$), то LTS либо выполняет один из переходов $s—z\rightarrow s`$, где $z\in P_{\gamma\tau}$, выбираемый недетерминированным образом, либо стоит, если таких переходов нет.
- Выполнение перехода $s—z\rightarrow s`$ происходит за конечное ненулевое время, после чего LTS оказывается в состоянии $s`$. При выполнении такого

перехода LTS сообщает машине о выполнении действия z, если $z \in P_\gamma$. О выполнении τ-переходов LTS ничего не сообщает машине.[19]

- Если LTS останавливается, её состояние s не меняется. LTS сообщает машине о своей остановке, если была нажата кнопка "P" с наблюдаемым отказом, то есть $P \in \mathfrak{R}$. Если была нажата кнопка "Q" с ненаблюдаемым отказом, то есть $Q \in \mathfrak{Q}$, или не была нажата кнопка, LTS не сообщает машине о своей остановке.[20]
- Для моделирования дивергенции будем считать, что каждый раз, когда LTS оказывается в дивергентном состоянии и нажата какая-нибудь кнопка, она может сообщить, а может и не сообщить, машине о выполнении Δ-действия. Выбор «сообщать – не сообщать» делается недетерминированным образом.[21] Если ни одна кнопка не нажата, LTS не сообщает машине о выполнении Δ-действия.

□

Теорема 11: $\mathfrak{R}/\mathfrak{Q}$-машина, внутри «чёрного ящика» которой находится LTS-модель $\mathbf{s}$ с правилами работы, согласно Определению 47, работает правильно.

□456

4.5. Эквивалентность трассовой и LTS-моделей

Структура раздела:

1. Преобразование трассовой модели в LTS-модель
2. «Компактное» преобразование
3. Утверждение об эквивалентности

---

[19] Это эквивалентно тому, что реализация сообщает машине о любом выполняемом действии, но машина игнорирует сообщение о выполнении τ-действия.

[20] Это эквивалентно тому, что LTS сообщает машине о любой своей остановке, но машина игнорирует сообщение, если нажата $\mathfrak{Q}$-кнопка или никакая кнопка не нажата.

[21] Это эквивалентно тому, что реализация всегда сообщает машине о дивергенции, но машина недетерминированным образом игнорирует или не игнорирует это сообщение.

Для завершения «круга эквивалентности» (Рис.25) машины тестирования, трассовой и LTS-моделей нам осталось показать, что для каждой трассовой $\mathfrak{R}$-модели существует LTS, имеющая эту модель в качестве множества своих $\mathfrak{R}$-трасс (Рис.39). Достаточно доказать это утверждение для ***F***-моделей и ***F***-трасс LTS.

Для трассовой $\mathfrak{R}$-модели существует LTS с тем же множеством $\mathfrak{R}$-трасс

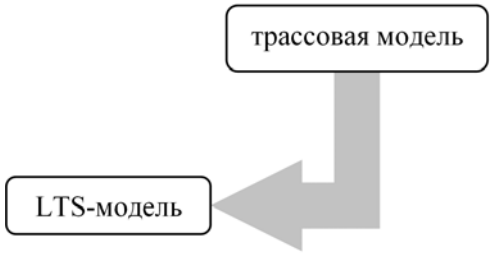


Рис.39.

4.5.1. Преобразование трассовой модели в LTS-модель

Сначала определим преобразование ***F2L*** : ***F***-модель→LTS-модель, в котором состояниями LTS будут ***F***-трассы заданной ***F***-модели (Рис.40).

Преобразование трассовой модели в LTS-модель

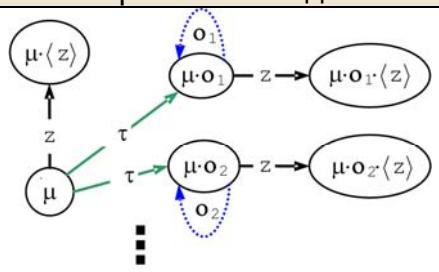


Рис.40.

τ-переход ведёт из ***F***-трассы μ, не заканчивающейся на отказ, в ***F***-трассу μ·o, заканчивающуюся на непустую трассу отказов o. Переход по внешнему действию z ведёт из состояния, соответствующего ***F***-трассе σ, в состояние,

соответствующее ***F***-трассе $\sigma\cdot\langle z\rangle$. Δ-переход заменяется на τ-петлю. LTS конечна тогда и только тогда, когда ***F***-модель (как множество ***F***-трасс) конечна.

Определение 48: Для алфавита L определим преобразование $\boldsymbol{F2L}:\boldsymbol{FMODEL}(L)\to\boldsymbol{LTS}(L_\gamma)$. Для $\Sigma\in\boldsymbol{FMODEL}(L)$ определим $\boldsymbol{F2L}(\Sigma)=\mathbf{s}=LTS(\Sigma,L_\gamma,E_{\mathbf{s}},\epsilon)$, где $E_{\mathbf{s}}$ – наименьшее множество, порождаемое следующими правилами вывода: $\forall\mu\in\Sigma\ \ \forall z\in L_\gamma\ \ \forall o\in\mathcal{P}(L)^*$

(1) $z\in L_\gamma\ \&\ \mu\cdot\langle z\rangle\in\Sigma \quad\vdash\ \mu\text{—}z\to\mu\cdot\langle z\rangle$;

(2) $\boldsymbol{postf}(\mu)=\epsilon\ \&\ o\neq\epsilon\ \&\ \mu\cdot o\in\Sigma \quad\vdash\ \mu\text{—}\tau\to\mu\cdot o$;

(3) $\boldsymbol{postf}(\mu)=\epsilon\ \&\ \mu\cdot\langle\Delta\rangle\in\Sigma \quad\vdash\ \mu\text{—}\tau\to\mu$.

□

Сначала докажем два вспомогательных утверждения о LTS $\boldsymbol{F2L}(\Sigma)$.

Лемма 10: Пусть $\Sigma\in\boldsymbol{FMODEL}(L)$ и $\mathbf{s}=\boldsymbol{F2L}(\Sigma)$. В LTS $\mathbf{s}$ достижимое состояние стабильно тогда и только тогда, когда оно соответствует ***F***-трассе σ, которая заканчивается либо разрушением, $\sigma=\mu\cdot\langle\gamma\rangle\in\Sigma$ (терминальное состояние), либо отказом $\sigma=\mu\cdot o\in\Sigma$, где $o\neq\epsilon$, $o\in\mathcal{P}(L)^*$. В любом из этих случаев $\boldsymbol{init}(\sigma)=\boldsymbol{head}(\Sigma\ \boldsymbol{after}\ \sigma)\cap L$. Кроме того, достижимое состояние конвергентно тогда и только тогда, когда оно соответствует ***F***-трассе, не продолжающейся в Σ Δ-действием.

□456

Лемма 11: Пусть $\Sigma\in\boldsymbol{FMODEL}(L)$ и $\mathbf{s}=\boldsymbol{F2L}(\Sigma)$. Тогда для каждого маршрута $S\in\boldsymbol{R}(\mathbf{s})$ множество его ***F***-трасс $\boldsymbol{F}(S)=\boldsymbol{drti}_{\mathrm{in}\,\Sigma^\circ}\omega(S)\cup(\{\omega(S)\cdot\langle\Delta\rangle\}\cap\Sigma)$.

□457

Теперь докажем основное утверждение о LTS ***F2L***(**Σ**).

Теорема 12: Пусть **Σ**∈***FMODEL***(L) и **s**=***F2L***(**Σ**). Тогда **Σ**=***F***(**s**).

□459

4.5.2. «Компактное» преобразование

Преобразование ***F2L*** создаёт очень «некомпактную» LTS. Вместо этого можно делать преобразование, используя разложение ***F***-модели на стабильные и контрстабильные поддеревья.

В Теорема 6: показано, что каждая ***F***-модель **Σ** разлагается в объединение наибольшего контрстабильного поддерева Σ_C и множества Σ_S стабильных поддеревьев. Мы будем строить LTS, в которой начальное состояние соответствует всей ***F***-модели **Σ** (Рис.41).

«Компактное» преобразование трассовой модели в LTS-модель

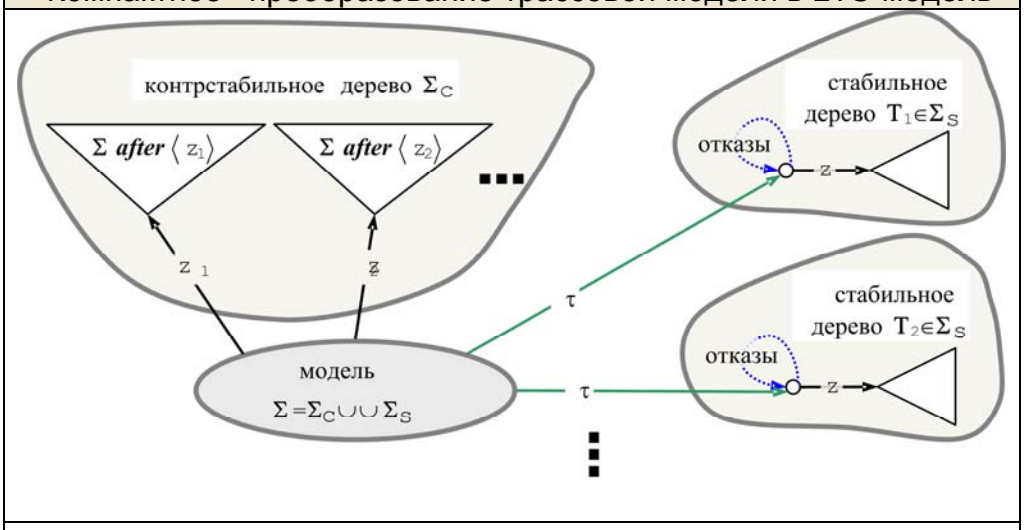


Рис.41.

Из начального состояния проводим τ-переходы в стабильные состояния, соответствующие всем стабильным поддеревьям из Σ_S. Если хотя бы одна ***F***-трасса из **Σ** начинается с некоторого внешнего действия z, проводим из начального состояния переход по z в состояние, соответствующее ***F***-модели **Σ** ***after*** ⟨z⟩. Если есть трасса ⟨γ⟩ делаем переход в состояние,

соответствующее множеству $\mathbf{\Sigma}$ ***after*** $\langle\gamma\rangle$ = $\{\epsilon\}$. Если есть трасса $\langle\Delta\rangle$ делаем τ-петлю в состоянии $\mathbf{\Sigma}$. Совокупность таких переходов по базовым действиям и продолжающих их (после внешнего действия) ***F***-моделей соответствует наибольшему контрстабильному поддереву $\mathbf{\Sigma}_c$. Разлагая ***F***-модель на каждом шаге, мы получаем искомую LTS.

Формально состояниями такой LTS объявим все возможные ***F***-модели в базовом алфавите, однако, множество достижимых состояний «минимально». Такое преобразование создаёт максимально «компактную» LTS. В частности, если для данной ***F***-модели существует конечная LTS с таким же множеством ***F***-трасс, то таким преобразованием будет получена конечная LTS.

Определение 49: Для алфавита L определим преобразование $\boldsymbol{F2_{compact}L}: \boldsymbol{FMODEL}(L) \rightarrow \boldsymbol{LTS}(L_\gamma)$. Для $\mathbf{\Sigma} \in \boldsymbol{FMODEL}(L)$ определим $\boldsymbol{F2_{compact}L}(\mathbf{\Sigma}) = \mathbf{S} = LTS(V_{\mathbf{S}}, L_\gamma, E_{\mathbf{S}}, \mathbf{\Sigma})$, где $V_{\mathbf{S}} = \{\{\epsilon\}\} \cup \boldsymbol{FMODEL}(L)$, а $E_{\mathbf{S}}$ – наименьшее множество, порождаемое следующими правилами вывода: $\forall \mathbf{T} \in \boldsymbol{FMODEL}(L)$

(1) $z \in \boldsymbol{head}(\mathbf{T}) \cap L_\gamma \quad \vdash \mathbf{T} \text{—} z \rightarrow \mathbf{T}'$ ***after*** $\langle z \rangle$,

(2) $\mathbf{T}$ ***unstable*** & $\mathbf{T} = \mathbf{T}_c \cup \cup \mathbf{T}_s$ & $\mathbf{T}' \in \mathbf{T}_s \quad \vdash \mathbf{T} \text{—} \tau \rightarrow \mathbf{T}'$,

(3) $\Delta \in \boldsymbol{head}(\mathbf{T}) \quad \vdash \mathbf{T} \text{—} \tau \rightarrow \mathbf{T}$,

где $\mathbf{T} = \mathbf{T}_c \cup \cup \mathbf{T}_s$ – разложение ***F***-модели по Теорема 6:.

□

Сначала докажем вспомогательное утверждение о стабильных и конвергентных состояниях LTS $\boldsymbol{F2_{compact}L}(\mathbf{\Sigma})$.

Лемма 12: Пусть $\mathbf{\Sigma} \in \boldsymbol{FMODEL}(L)$ и $\mathbf{S} = \boldsymbol{F2_{compact}L}(\mathbf{\Sigma})$. В LTS $\mathbf{S}$ состояние стабильно тогда и только тогда, когда это терминальное состояние $\{\epsilon\}$ или стабильное дерево $\mathbf{T}$. В любом из этих случаев $\boldsymbol{init}(\mathbf{T}) = \boldsymbol{head}(\mathbf{T}) \cap L$. Кроме

того, состояние конвергентно тогда и только тогда, когда оно соответствует дереву, в котором нет трассы $\langle\Delta\rangle$.

□461

Теперь докажем основное утверждение о LTS $F2_{compact}L(\Sigma)$.

Теорема 13: Пусть $\Sigma\in FMODEL(L)$ и $S=F2_{compact}L(\Sigma)$. Тогда $\Sigma=F(S)$.

□462

4.5.3. Утверждение об эквивалентности

Теорема 9: вместе с Теорема 12: или Теорема 13: дают следующее утверждение об эквивалентности моделей.

Теорема 14: F-модели и LTS-модели в алфавите L эквивалентны в следующем смысле: $F\circ LTS(L_\gamma) = FMODEL(L)$.

□466

Для $\mathfrak{R}$-моделей мы также имеем:

$$\{F(S)^{\downarrow L_{\mathfrak{R}\Delta\gamma}} \mid S\in LTS(L_\gamma)\} = MODEL(L_{\mathfrak{R}\Delta\gamma}).$$

Одной $\mathfrak{R}$-модели соответствует, вообще говоря, множество имеющих то же множество $\mathfrak{R}$-трасс LTS-моделей (Рис.42).

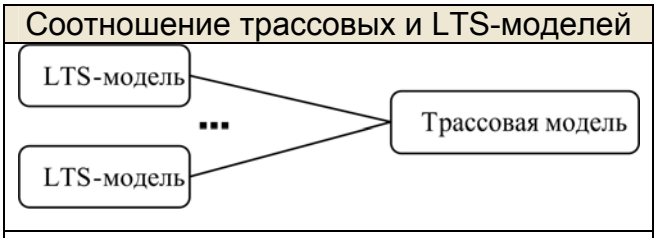


Рис.42.

4.6. Безопасность и конформность

Опираясь на эквивалентность трассовой и LTS-моделей, мы определим гипотезу о безопасности и безопасную конформность для LTS, заимствуя соответствующие определения из трассовой теории.

Определение 50:

- LTS-реализацией будем называть LTS-модель $\mathbf{I} \in \boldsymbol{LTS}(L_\gamma)$.
- LTS-спецификацией будем называть LTS-модель $\mathbf{S} \in \boldsymbol{LTS}(L_\gamma)$, если задан протокол взаимодействия с помощью отношения ***safe by*** на ***F***-модели $\boldsymbol{F}(\mathbf{S})$.

□

Определение 51:

- Будем говорить, что LTS-реализация **I** *безопасна* для LTS-спецификации **S**, если ***F***-модель реализации безопасна для ***F***-модели спецификации:

  $\mathbf{I}$ ***safe for*** $\mathbf{S} =_{\text{def}} \boldsymbol{F}(\mathbf{I})$ ***safe for*** $\boldsymbol{F}(\mathbf{S})$.
- Класс LTS-реализаций, безопасных для данной LTS-спецификации, обозначим:

  $\boldsymbol{safe\mathfrak{I}}(\mathbf{S}) =_{\text{def}} \{\mathbf{I} \in \boldsymbol{LTS}(L_\gamma) \mid \mathbf{I}$ ***safe for*** $\mathbf{S}\}$.

□

Заметим, что класс безопасных LTS-реализаций не является множеством, поскольку мы не налагаем никаких ограничений на их состояния.

Определение 52:

- Будем говорить, что LTS-реализация **I** *безопасно конформна* (или, для краткости, просто *конформна*) LTS-спецификации **S**, если ***F***-модель реализации безопасно-конформна ***F***-модели спецификации:

  $\mathbf{I}$ ***saco*** $\mathbf{S} =_{\text{def}} \boldsymbol{F}(\mathbf{I})$ ***saco*** $\boldsymbol{F}(\mathbf{S})$.
- Класс реализаций, конформных спецификации **S**, обозначим:

  $\mathfrak{I}(\mathbf{S}) =_{\text{def}} \{\mathbf{I} \in \boldsymbol{LTS}(L_\gamma) \mid \mathbf{I}$ ***saco*** $\mathbf{S}\}$.

□

Заметим, что класс конформных LTS-реализаций не является множеством, поскольку мы не налагаем никаких ограничений на их состояния.

4.7. Композиция LTS и тесты

Структура раздела:

1. Параллельная композиция LTS
2. LTS-тесты
3. Классификация LTS по ветвимости
4. Алгоритмизация

Для LTS-модели мы определили безопасные (удовлетворяющие гипотезе о безопасности, то есть отношению ***safe for***) реализации и безопасную конформность ***saco***, используя эквивалентность трассовой модели и LTS-модели и соответствующие определения трассовой теории. Для LTS-модели взаимодействие моделируется оператором параллельной композиции алгебры процессов. При тестировании компонуются LTS-реализация и LTS-тест, результатом становится новая, композиционная LTS, состояния которой – это пары 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 $\mathtt{i}$ состояние реализации.

Используя LTS-отношение ***passes***, мы могли бы определить значимые, исчерпывающие и полные наборы LTS-тестов аналогично тому, как это делается в трассовой теории, используя отношение ***passes*** для трассовых моделей. Вместо этого мы определим преобразование ***Tt2Lt*** трассовых тестов в LTS-тесты. С одной стороны, мы имеем трассовое отношение ***passes*** между трассовой реализацией $\boldsymbol{F}(\mathtt{I})$ и трассовым тестом $\mathbf{T} \in \boldsymbol{Ttests}(\boldsymbol{F}(\mathtt{S}), \mathfrak{R}, \mathfrak{Q})$. С другой стороны, мы имеем LTS-отношение ***passes*** между LTS-реализацией $\mathtt{I}$ и LTS-тестом $\boldsymbol{Tt2Lt}(\mathbf{T})$. Мы покажем, что эти отношения эквивалентны (Рис.43). Тем самым, будет показано, что множество $\boldsymbol{Tt2Lt}(\mathbf{TT})$ LTS-тестов, где $\mathbf{TT} \subseteq \boldsymbol{Ttests}(\boldsymbol{F}(\mathtt{S}), \mathfrak{R}, \mathfrak{Q})$ полный набор трассовых тестов для трассовой

спецификации $\boldsymbol{F}(\mathbf{S})$, является полным набором LTS-тестов для LTS-спецификации $\mathbf{S}$.

Эквивалентность отношений *passes* в трассовой теории и LTS-теории

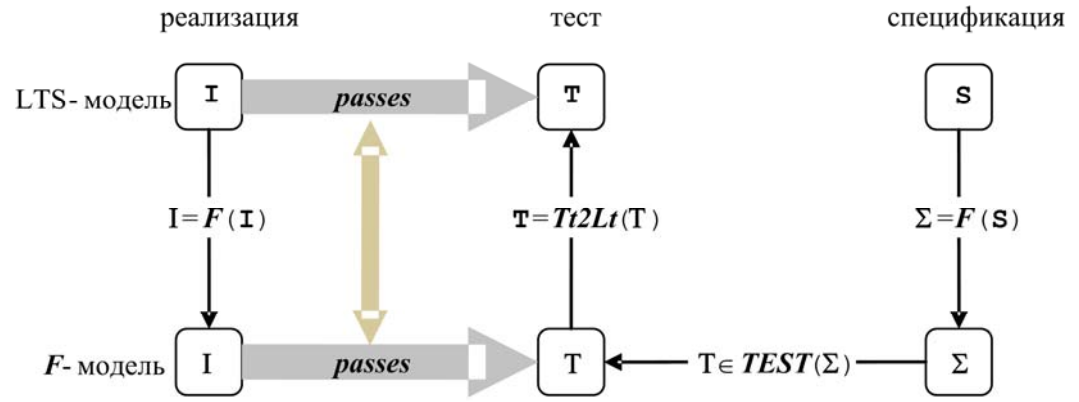


Рис.43.

4.7.1. Параллельная композиция LTS

Определение 53: Будем считать, что на универсуме внешних действий Z задана инволюция (биекция, обратная сама себе) «подчёркивание», которая каждому внешнему действию z ставит в соответствие *противоположное* действие $\underline{z}$ так, что $\underline{\underline{z}}=z$. Мы распространим эту инволюцию на разрушение и дивергенцию: $\underline{\gamma}=\gamma$, $\underline{\Delta}=\Delta$, а также множества базовых символов: $\forall U\subseteq Z_{\Delta\gamma}\ \underline{U}=\{\underline{u}\,|\,u\in U\}$.

□

Примером могут служить реактивные системы, в которых стимулы и реакции противоположны друг другу. Реализация принимает свой стимул $?x$ тогда, когда окружение выдаёт его как свою реакцию $!x$; реализация выдаёт реакцию $!y$ тогда, когда окружение принимает её как свой стимул $?y$.

Параллельное выполнение двух LTS в алфавитах A и B понимается так, что переходы по противоположным действиям z и $\underline{z}$, где $z\in A$ и $\underline{z}\in B$, выполняются синхронно, то есть, в обеих LTS одновременно, причём в композиции это становится τ-переходом. Переходы в LTS-операндах по символу τ выполняются асинхронно, то есть, в одной из LTS при сохранении

состояния другой LTS. Это соответствует определению параллельной композиции в CCS – Calculus of Communicating Systems [122] (также [105]).

Заметим, что при параллельной композиции в CSP – Communicating Sequential Processes [101] синхронно выполняется пара переходов по одному и тому же внешнему действию z, который становится символом композиционного перехода. Далее требуется дополнительное применение оператора «сокрытия» ***Hide*** [66], который превращает композиционные переходы, соответствующие синхронной паре переходов в компонентах, в τ-переходы. В общем, параллельная композиция в CCS и CSP (вместе с «сокрытием») аналогичны друг другу, за исключением того, что в CSP возможно «широковещание», когда посылаемый стимул может быть одновременно принят несколькими компонентами композиционной системы. В данной работе мы не рассматриваем такого «широковещания».

Поскольку мы ввели в LTS новый тип перехода – переход по разрушению, мы должны определить, как выполняются такие переходы при композиции LTS. Будем считать, что γ-переходы выполняются асинхронно – так же как τ-переходы. Это соответствует семантике разрушения, как нежелательного поведения, которое может происходить независимо от окружения (не может запрещаться кнопками машины тестирования). Если разрушается один компонент, то система в целом считается разрушенной.

Фиксация $\mathfrak{R}\backslash\mathfrak{Q}$-семантики означает, что в каждом состоянии LTS-теста множество R тех внешних действий, по которым определены переходы из этого состояния, противоположно одному из кнопочных множеств, то есть $\underline{R}\in\mathfrak{R}\cup\mathfrak{Q}$. Наблюдение отказа реализуется в LTS-тесте с помощью θ-перехода [111,149], который предназначен для разрешения тупиков и происходит тогда, когда никакие другие переходы невозможны. Поскольку не всякий отказ наблюдаем, а только тот, который соответствует $\mathfrak{R}$-кнопке “P”, в соответствующем состоянии теста, кроме θ-перехода, должны быть определены переходы по каждому действию $z\in\underline{P}$, и только они.

Определение 54: Будем считать, что в рамках данной теории среди *праэлементов* (объектов, не являющихся классами) выделен символ θ так же, как символы Δ, γ, τ и внешние действия.

□

Мы не будем вводить двух обозначений для оператора параллельной композиции LTS без θ-переходов и с θ-переходами (как это делает Тритманс в [149], используя обозначения $\|$ и $\rceil\|$), считая, что мы просто расширяем оператор параллельной композиции для LTS с θ-переходами. Мы будем использовать обозначение $\rceil\lfloor$, чтобы подчеркнуть, что композиция происходит в духе алгебры процессов CCS.

Определение 55: Для алфавитов внешних действий определим их композицию $.\rceil\lfloor.: \wp(Z)\times\wp(Z)\to\wp(Z)$ следующим образом:

$\forall A,B\subseteq Z\ A\rceil\lfloor B =_{def} (A\setminus\underline{B})\cup(B\setminus\underline{A})$.

- 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 действиями.
- 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 внешними действиями.
- τ-, Δ- и γ-действия также будем называть асинхронными действиями.

□

Композиция алфавитов является коммутативной операцией. Заметим, что Δ-действий в LTS нет, они появляются только в трассовой модели LTS для моделирования дивергенции. Тем не менее, их естественно считать асинхронными в том смысле, что композиционное состояние it всегда дивергентно, если дивергентно состояние одного из операндов: i или t.

Теперь определим формально (параллельную) композицию LTS. Для краткости мы будем писать элемент декартового произведения как it вместо (i,t).

Определение 56: Пусть заданы алфавиты $A,B\subseteq Z$.

Определим композицию $.|\!|.: \boldsymbol{LTS}(A_{\gamma\theta})\times\boldsymbol{LTS}(B_{\gamma\theta})\to\boldsymbol{LTS}(C_{\gamma})$, где $C=A|\!|B$, следующим образом: $\forall \mathbf{I}\in\boldsymbol{LTS}(A_{\gamma\theta})\ \forall \mathbf{T}\in\boldsymbol{LTS}(B_{\gamma\theta})$

$\mathbf{I}|\!|\mathbf{T} =_{def} LTS(V_{\mathbf{I}}\times V_{\mathbf{T}}, C_{\gamma}, E, i_0t_0)$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall i\in V_{\mathbf{I}}\ \forall t\in V_{\mathbf{T}}$

$(|\!|1)\quad z\in A_{\gamma\tau}\setminus\underline{B} \ \&\ i\text{—}z\to i` \ \vdash it\text{—}z\to i`t,$

$(|\!|2)\quad z\in B_{\gamma\tau}\setminus\underline{A} \ \&\ t\text{—}z\to t` \ \vdash it\text{—}z\to it`,$

$(|\!|3)\quad z\in A\cap\underline{B} \ \&\ i\text{—}z\to i` \ \&\ t\text{—}\underline{z}\to t` \ \vdash it\text{—}\tau\to i`t`,$

$(|\!|4)\quad t\text{—}\theta\to t` \ \&\ \boldsymbol{Deadlock}(i,t) \ \vdash it\text{—}\tau\to it`,$

$(|\!|5)\quad i\text{—}\theta\to i` \ \&\ \boldsymbol{Deadlock}(i,t) \ \vdash it\text{—}\tau\to i`t,$

где $\boldsymbol{Deadlock}(i,t) = i\text{—}\tau\gamma\not\to \ \&\ t\text{—}\tau\gamma\not\to$

$\&\ (\forall z\in A\cap\underline{B}\ i\text{—}z\not\to \vee t\text{—}\underline{z}\not\to).$

□

Композиция LTS является коммутативной операцией (с точностью до изоморфизма, т.е. именования состояний it или ti).

Композиция двух LTS естественно распространяется на композицию класса LTS и одной LTS, одной LTS и класса LTS, и двух классов LTS как класс попарных композиций.

Мы допускаем θ-переходы только в тесте. Если реализация представляет собой составную систему, скомпонованную из нескольких LTS-компонентов, то в этих компонентах нет θ-переходов также, как в композиционной LTS-реализации. В Глава 2 мы говорили о том, что допущение θ-переходов в

реализации означало бы введение приоритетов, что было бы полезно для решения многих проблем. В этом случае, при многократной композиции реализационных компонентов, по-видимому, не всякий θ-переход должен превращаться в τ-переход. Только в окончательной композиции с тестом (полным окружением), когда получается замкнутая система в пустом алфавите, все θ-переходы должны исчезнуть. Это означает наличие θ-переходов разных типов. В CSP θ-переходы только одного типа скрывались бы одним применением оператора ***Hide***. Вопросы расширения теории на случай систем с приоритетами выходят за рамки темы данной работы и в дальнейшем мы не будем на этом останавливаться.

4.7.2. LTS-тесты

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). Алфавит композиции реализации и теста пуст: $L \rfloor\!\lfloor \underline{L} = \varnothing$. Поэтому возможны лишь τ- и γ-переходы. Если тестирование безопасно в заданной $\mathfrak{R}\backslash\mathfrak{Q}$-семантике, то в композиции нет γ-переходов и дивергенции. В этом случае взаимодействие теста и реализации означает прохождение некоторого τ-маршрута в композиции LTS-реализации и LTS-теста. При этом реализация проходит маршрут с трассой в алфавите L, а тест проходит маршрут с трассой в алфавите $\underline{L}_\theta$. В терминах $\mathfrak{R}$-маршрутов реализации мы можем считать, что при прохождении тестом θ-перехода реализация проходит переход-петлю по $\mathfrak{R}$-отказу. Тем самым, можно считать, что реализация проходит $\mathfrak{R}$-маршрут с $\mathfrak{R}$-трассой σ, то есть трассу в алфавите $L_{\mathfrak{R}}$.

Мы будем рассматривать LTS-тест как строго-конвергентную LTS с θ-переходами, без разрушения. Вердикт в LTS-тесте определяется как

терминальное состояние ***pass*** или ***fail***, других терминальных состояний не должно быть.

Поскольку мы рассматриваем тестирование без приоритетов, мы можем запретить переключение кнопок. Поэтому не должно быть совмещения переходов по внешним действиям или θ-переходов с τ-переходами: в каждом нестабильном состоянии определены только τ-переходы. С другой стороны, наличие τ-переходов делает LTS-тест недетерминированным так же, как возможность нескольких θ-переходов или переходов по одному внешнему действию из одного состояния. Понятно, что такой недетерминизм не увеличивает мощности тестирования: вместо одного недетерминированного теста мы могли бы иметь несколько детерминированных тестов. Каждый такой детерминированный тест получается из исходного недетерминированного теста следующей (недетерминированной) процедурой: 1) если в начальном состоянии есть τ-переход $s_0 \text{—}\tau\text{→} s$, этот переход удаляется, а в качестве нового начального состояния выбирается состояние s_0 или состояние s; этот шаг повторяется до тех пор, пока в начальном состоянии не останется τ-переходов; 2) любые два перехода $s \text{—}u\text{→} s` \text{—}\tau\text{→} s``$ заменяются на один переход $s \text{—}u\text{→} s``$; 3) в каждом состоянии s выбирается только одного из нескольких переходов вида $s \text{—}u\text{→} s`$ для каждого $u \in \underline{L}_\theta$, а остальные удаляются. В дальнейшем мы ограничиваемся только детерминированными LTS-тестами, в которых все состояния стабильны.

Поскольку тест определяется для заданной $\mathfrak{R}\backslash\mathfrak{Q}$-семантики, в каждом нетерминальном состоянии множество внешних действий, противоположных тем, которыми помечены исходящие переходы, совпадает с некоторым отказом $P \in \mathfrak{R} \cup \mathfrak{Q}$, а θ-переход определён тогда и только тогда, когда $P \in \mathfrak{R}$. Такое состояние будем называть P-состоянием. Мы определим преобразование, которое заменяет θ-переход из P-состояния, где $P \in \mathfrak{R}$, на переход по отказу P,

а внешние действия заменяет на противоположные. Для маршрута, заканчивающегося в P-состоянии, кнопка “P” должна быть допустима после преобразованной трассы этого маршрута.

Условие конечности времени выполнения для LTS-теста сводится к отсутствию бесконечных маршрутов. Для детерминированного теста это эквивалентно нётеровости дерева (преобразованных) трасс.

Если взять преобразованные трассы такого LTS-теста, то мы получим трассовый тест. При этом вердикты нужно согласовать: вердикт, назначенный в трассовом тесте максимальной трассе, совпадает с вердиктом в конце маршрута LTS-теста, имеющего (после преобразования) эту трассу.

Композиция LTS-теста с LTS-реализацией, удовлетворяющей гипотезе о безопасности, содержит только конечные τ-маршруты, а все достижимые терминальные состояния имеют вид i***fail*** или i***pass***. Выполнение такой композиционной LTS завершается за конечное время. LTS-реализация *проходит* LTS-тест, если в композиции реализации и теста недостижимы состояния вида i***fail***.

В общем, любой такой LTS-тест можно получить из трассового теста с помощью преобразования ***Tt2Lt***. Состояниями LTS-теста будут немаксимальные тестовые трассы и два терминальных состояния ***pass*** и ***fail***. Переход по внешнему действию соответствует продолжению трассы этим действием, а θ-переход – продолжению трассы отказом.

Определение 57: Для алфавита L определим преобразование $\underline{L}_\theta 2L_{\mathcal{P}(L)} : LTS(\underline{L}_\theta) \to LTS(L_{\mathcal{P}(L)})$. Для $\mathbf{T}=LTS(V_T, \underline{L}_\theta, E_T, t_0)$ $\underline{L}_\theta 2L_{\mathcal{P}(L)}(\mathbf{T})=LTS(V_T, L_{\mathcal{P}(L)}, E, t_0)$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall t,t' \in der(\mathbf{T})$

$\underline{z} \in \underline{L}$ & $t \xrightarrow{\underline{z}} t'$ $\vdash$ $t \xrightarrow{z} t'$;

$t \xrightarrow{\theta} t'$ $\vdash$ $t \xrightarrow{L \setminus init(t)} t'$.

□

Определение 58: Пусть задана LTS-спецификация $\mathbf{S}$. Определим преобразование $\boldsymbol{Tt2Lt}:\boldsymbol{Ttests}(\boldsymbol{F}(\mathbf{S}),\mathfrak{R},\mathfrak{Q})\to\boldsymbol{LTS}(\underline{L}_\theta)$. Для $\mathbf{T}\in\boldsymbol{Ttests}(\boldsymbol{F}(\mathbf{S}),\mathfrak{R},\mathfrak{Q})$ $\boldsymbol{Tt2Lt}(\mathbf{T})=\mathrm{LTS}(V_T,\underline{L}_\theta,E_T,t_0)$, где $V_T=(\mathbf{T}\setminus\boldsymbol{max}(\mathbf{T}))\cup\{\boldsymbol{pass},\boldsymbol{fail}\}$, $t_0=\epsilon$, если $\mathbf{T}\neq\{\epsilon\}$, или $t_0=\boldsymbol{verdict}(\epsilon)$, если $\mathbf{T}=\{\epsilon\}$, а E_T – наименьшее множество, порождаемое следующими правилами вывода:

$\forall\sigma\in\mathbf{T}\setminus\boldsymbol{max}(\mathbf{T})\quad\forall P\in\mathfrak{R}\cup\mathfrak{Q}\quad\forall z\in L$

$\sigma\cdot\langle z\rangle\in\mathbf{T}\setminus\boldsymbol{max}(\mathbf{T})\quad\vdash\ \sigma\text{—}\underline{z}\to\sigma\cdot\langle z\rangle;$

$\sigma\cdot\langle z\rangle\in\boldsymbol{max}(\mathbf{T})\quad\vdash\ \sigma\text{—}\underline{z}\to\boldsymbol{verdict}(\sigma\cdot\langle z\rangle);$

σ P-трасса & $\sigma\cdot\langle P\rangle\in\mathbf{T}\setminus\boldsymbol{max}(\mathbf{T})\quad\vdash\ \sigma\text{—}\theta\to\sigma\cdot\langle P\rangle;$

σ P-трасса & $\sigma\cdot\langle P\rangle\in\boldsymbol{max}(\mathbf{T})\quad\vdash\ \sigma\text{—}\theta\to\boldsymbol{verdict}(\sigma\cdot\langle P\rangle).$

□

Определение 59: Пусть задана LTS-спецификация $\mathbf{S}$.

- *LTS-тестом или тестовым примером (test case)* будем называть LTS, полученную из трассового теста преобразованием ***Tt2Lt***. Множество всех LTS-тестов обозначим
  $\boldsymbol{Ltests}(\mathbf{S},\mathfrak{R},\mathfrak{Q})=_{\text{def}}\boldsymbol{Tt2Lt}\circ\boldsymbol{Ttests}(\boldsymbol{F}(\mathbf{S}),\mathfrak{R},\mathfrak{Q})$.
- *Набор LTS-тестов (test suite)* – это множество LTS-тестов: $\mathbf{TT}\subseteq\boldsymbol{Ltests}(\mathbf{S},\mathfrak{R},\mathfrak{Q})$.
- Для LTS-реализации $\mathbf{I}\in\boldsymbol{LTS}(L_\gamma)$ и LTS-теста $\mathbf{T}\in\boldsymbol{Ltests}(\mathbf{S},\mathfrak{R},\mathfrak{Q})$

  $\mathbf{I}$ ***passes*** $\mathbf{T}$ $=_{\text{def}}$ в LTS $\mathbf{I}\pitchfork\mathbf{T}$ $\forall it\in\boldsymbol{der}(i_0t_0)$ $t\neq\boldsymbol{fail}$.

□

Теорема 15: Преобразование $\boldsymbol{traces}\circ\underline{\boldsymbol{L}}_\theta\boldsymbol{2L}_{\mathcal{P}(L)}$ обратно преобразованию ***Tt2Lt***.

Пусть задана LTS-спецификация $\mathbf{S}\in\boldsymbol{LTS}(L_\gamma)$. Тогда:

1) $\forall \mathrm{T} \in \mathit{Ttests}(\mathit{F}(\mathbf{s}), \mathfrak{R}, \mathfrak{Q})\ \ \mathit{traces} \circ \underline{\mathit{L}}_{\theta} \mathit{2L}_{\mathcal{P}(\mathrm{L})} \circ \mathit{Tt2Lt}(\mathrm{T}) = \mathrm{T}$;

2) вердикты согласованы:

$\mathit{max}(\mathrm{T}) = \{\mathrm{s} \in \mathit{der} \circ \underline{\mathit{L}}_{\theta} \mathit{2L}_{\mathcal{P}(\mathrm{L})} \circ \mathit{Tt2Lt}(\mathrm{T}) \mid \mathrm{s}$ терминально$\}$

$= \{\mathrm{s} \in \mathit{der} \circ \mathit{Tt2Lt}(\mathrm{T}) \mid \mathrm{s}$ терминально$\}$,

$\forall \sigma \in \mathit{max}(\mathrm{T})\ \ \underline{\mathit{L}}_{\theta} \mathit{2L}_{\mathcal{P}(\mathrm{L})} \circ \mathit{Tt2Lt}(\mathrm{T})\ \ \mathit{after}\ \ \sigma = \{\mathit{verdict}(\sigma)\}$.

□466

Из Теорема 15: следует:

$\mathit{Ttests}(\mathit{F}(\mathbf{s}), \mathfrak{R}, \mathfrak{Q}) = \mathit{traces} \circ \underline{\mathit{L}}_{\theta} \mathit{2L}_{\mathcal{P}(\mathrm{L})} \circ \mathit{Ltests}(\mathbf{s}, \mathfrak{R}, \mathfrak{Q})$.

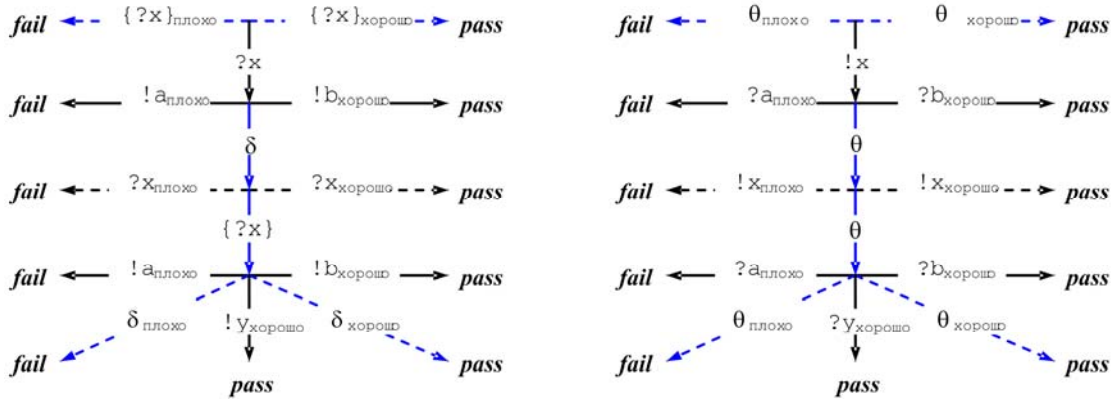
Теорема 16: Пусть заданы LTS-спецификация $\mathbf{s}$, безопасная LTS-реализация $\mathbf{I} \in \mathit{safe}\mathfrak{I}(\mathbf{s})$ и трассовый тест $\mathrm{T} \in \mathit{Ttests}(\mathit{F}(\mathbf{s}), \mathfrak{R}, \mathfrak{Q})$.

Тогда $\mathbf{I}\ \ \mathit{passes}\ \ \mathit{Tt2Lt}(\mathrm{T}) \Leftrightarrow \mathit{F}(\mathbf{I})\ \ \mathit{passes}\ \ \mathrm{T}$.

□467

В LTS-тесте, построенном по трассовому примитивному тесту, все маршруты имеют ограниченную длину, что соответствует ограниченности времени выполнения теста. На Рис.44 показан пример генерации LTS-теста по примитивному трассовому тесту, который строится по одной тестовой трассе спецификации.

Генерация примитивного трассового теста и примитивного LTS-теста для тестовой трассы $\delta=\langle ?x,\delta,\{?x\},!y\rangle$ в $\beta\gamma\delta$-семантике



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 трассового теста и из одного состояния LTS-теста.

Рис.44.

4.7.3. Классификация LTS по ветвимости

Для дальнейшего нам понадобится определение целого ряда классов LTS, отличающихся теми или иными ограничениями на ветвимость. Хотя некоторые из этих классов понадобятся много ниже по тексту, удобнее свести их здесь в единую систему (мы определим только те классы, которые нам понадобятся, не претендуя на полную классификацию ветвимости LTS-моделей).

Определение 60:

- *Безопасно-достижимым* или *Б-достижимым состоянием* LTS-модели будем называть состояние, достижимое (из начального состояния) по безопасной $\mathfrak{R}$-трассе.
- *Трансбезопасно-достижимым* или *ТБ-достижимым состоянием* LTS-модели будем называть начальное состояние, Б-достижимое состояние или постсостояние перехода из Б-достижимого состояния.
- *Неразрушающим состоянием* LTS-модели будем называть такое достижимое состояние s, в котором нет γ-трассы: $s=\langle\gamma\rangle\not\Rightarrow$.

□

Трансбезопасное состояние – это состояние, достижимое по пустой трассе или безопасной $\mathfrak{R}$-трассе, продолженной одним действием.

Определение 61:

- Состояние LTS-модели будем называть *конечно-ветвящимся* или *К-ветвящимся*, если в нём определено конечное число переходов суммарно по всем действиям (включая внутреннее действие τ).
- Состояние LTS-модели будем называть *локально-конечно-ветвящимся* или *ЛК-ветвящимся*, если в нём определено конечное число переходов по каждому действию (включая внутреннее действие τ).
- Состояние LTS-модели будем называть *безопасно-локально-конечно-ветвящимся* или *БЛК-ветвящимся*, если в нём определено конечное число переходов по каждому действию, безопасному после некоторой безопасной $\mathfrak{R}$-трассы, заканчивающейся в этом состоянии, а также конечное число τ-переходов.

□

Определение 62:

- Состояние LTS-модели будем называть *тау-ограниченным*, если из него по τ-маршрутам достижимо конечное число состояний.
- Состояние LTS-модели будем называть *тау-перечислимым*, если дерево τ-маршрутов, начинающихся в этом состоянии, перечислимо-ветвящееся.

□

Заметим, что К-ветвимое состояние может не быть тау-ограниченным, так как в состоянии может начинаться бесконечный τ-маршрут (например, τ-петля). Также очевидно, что тау-ограниченность влечёт тау-перечислимость, хотя обратное не верно. Тау-перечислимость состояния s эквивалентна тому, что во всех состояниях, достижимые из s по τ-маршрутам, определено перечислимое множество τ-переходов.

Определение 63: Определим некоторые классы LTS в зависимости от ветвимости их состояний и введём их сокращённые обозначения.

Обозначение LTS	Каждое
К-ветвящаяся	достижимое состояние К-ветвящееся
ЛК-ветвящаяся	достижимое состояние ЛК-ветвящееся
Н.ЛК-ветвящаяся	неразрушающее состояние ЛК-ветвящееся
Б.К-ветвящаяся	Б-достижимое состояние К-ветвящееся
Б.БЛК-ветвящаяся	Б-достижимое состояние БЛК-ветвящееся

□

Определение 64: Определим некоторые классы LTS в зависимости от тау-ограниченности и тау-перечислимости их состояний и введём их сокращённые обозначения.

Обозначение LTS	Каждое
О-ветвящаяся	достижимое состояние тау-ограниченное
Н.О-ветвящаяся	неразрушающее состояние тау-ограниченное
Б.О-ветвящаяся	Б-достижимое состояние тау-ограниченное
П-ветвящаяся	достижимое состояние тау-перечислимое
ТБ.П-ветвящаяся	ТБ-достижимое состояние тау-перечислимое

□

Определение 65: Введём сокращённые обозначения для некоторых смешанных классов LTS в зависимости от ветвимости, тау-ограниченности и тау-перечислимости их состояний.

	ЛК	Н.ЛК	Б.К	Б.БЛК
О	ЛК+О			
Б.О				Б.БЛК+Б.О
Н.О+П		Н.ЛК+Н.О+П		
Б.О+ТБ.П			Б.К+Б.О+ТБ.П	Б.БЛК+Б.О+ТБ.П

□

4.7.4. Алгоритмизация

Алгоритмическое задание LTS основано на локальных алгоритмах, которые по заданному состоянию определяют имеющиеся в нём переходы. Достаточно ограничиться только Б- и ТБ-достижимыми состояниями.

Прежде всего, нужно за конечное время определить, имеется ли в спецификации разрушение после $\mathfrak{R}$-трассы. Для этого нужно определить, есть ли среди состояний после $\mathfrak{R}$-трассы пресостояние γ-перехода. Поэтому, прежде всего, требуется 1) конечность множества состояний после $\mathfrak{R}$-трассы $\mathbf{s}$ ***after*** μ, если разрушения нет, и 2) перечислимость этого множества, если разрушение есть.

В случае 1) нам достаточно, чтобы $\mathfrak{R}$-трасса μ была безопасной, то есть нам требуется, чтобы каждая безопасная $\mathfrak{R}$-трасса заканчивалась в конечном множестве состояний.

Лемма 13: Б.БЛК+Б.О-ветвимость LTS-модели $\mathbf{s}$ является необходимым и достаточным условием того, чтобы каждая безопасная $\mathfrak{R}$-трасса μ заканчивалась в конечном множестве состояний $\mathbf{s}$ ***after*** μ.

□467

В случае 2) (перечислимость множества состояний $\mathbf{s}$ ***after*** μ, если в этом множестве есть пресостояние γ-перехода) достаточно, чтобы $\mathfrak{R}$-трасса μ была трансбезопасной (пустой, безопасной или безопасной трассой, продолженной одним действием). Также достаточно 2a) перечислимости множества состояний после пустой трассы и множества $\{s` \mid \exists s \in (\mathbf{s}\ \textit{after}\ \mu)\ s \text{—}z\rightarrow s`\}$ постсостояний переходов по одному внешнему действию из состояний после безопасной $\mathfrak{R}$-трассы, а также 2b) перечислимости дерева τ-маршрутов, начинающихся в каждом таком 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 должно быть П-ветвящимся, а не просто перечислимым[22]. Заметим, что нас интересует только сам факт наличия или отсутствия γ-перехода из состояния. Нам не важно, сколько есть таких переходов и в какие постсостояния они ведут. Таким образом, нам требуется, чтобы спецификация была ТБ.П-ветвящейся.

Итак, для алгоритмической генерации тестов требуется, чтобы LTS-спецификация была Б.БЛК+Б.О-ветвящаяся и ТБ.П-ветвящаяся, то есть Б.БЛК+Б.О+ТБ.П-ветвящаяся.

Если трасса безопасна, мы должны определить, продолжается она дивергенцией или нет. Поскольку число состояний после трассы конечно, дивергенция возможна только в виде τ-цикла, который легко обнаруживается алгоритмически. Дивергенция другого типа (τ-маршрут, проходящий через бесконечное число состояний) возможна только после такой трассы, которая продолжается разрушением, то есть опасна. В этом случае, используя свойство ТБ.П-ветвимости, мы можем обнаружить разрушение за конечное время. Заметим, что с точки зрения генерации тестов нам безразлично наличие или отсутствие трассы $\mu\cdot\langle\Delta\rangle$ при наличии трассы $\mu\cdot\langle\gamma\rangle$.

Другие ограничения связаны с вычислением безопасности кнопок и действий. Эти ограничения сильно зависят от семейств $\mathfrak{R}$ и $\mathfrak{Q}$ и способа их алгоритмического задания, способа алгоритмического задания отношения ***safe by*** (при наличии $\mathfrak{Q}$-кнопок), а также способа алгоритмического задания переходов в состояниях. Здесь мы не будем углубляться в эти вопросы. Отметим только, что в большинстве случаев при

---

[22] Из перечислимо-ветвимости дерева маршрутов следует его перечислимость, но не наоборот.

подходящем задании отношения ***safe by*** (или при отсутствии $\mathfrak{Q}$-кнопок) достаточно перечислимости множества кнопок $\mathfrak{R}\cup\mathfrak{Q}$ и конечности каждого кнопочного множества.

Алгоритмизация генерации LTS-тестов может быть основана на алгоритмизации генерации трассовых тестов и преобразований ***F*** и ***Tt2Lt***.

Преобразование ***F*** (взятие множества ***F***-трасс LTS) реализуется как преобразование алгоритмически заданной LTS (добавление петель отказов, γ- и Δ-переходов) и построение трасс маршрутов полученной LTS. Для алгоритмизуемости этих преобразований достаточно, чтобы 1) алфавит был конечен, 2) из каждого состояния вело только конечное число переходов по каждому действию и 3) любой бесконечный τ-маршрут являлся τ-циклом (проходил через конечное число состояний). Тогда любая трасса будет заканчиваться в конечном множестве состояний.

Преобразование ***Tt2Lt*** алгоритмизуется очевидным образом при алгоритмическом задании трассового теста. В частности, это так для конечного алфавита.

Отметим также, что преобразование ***F2L*** из трассовой модели в LTS легко алгоритмизуемо при конечном алфавите и некоторой модификации преобразования. Модификация заключается в следующем: если трасса μ продолжается двумя трассами отказов $\mathbf{o}_1$ и $\mathbf{o}_2$, причём одна из них получается из другой ***r***-операциями (вставка повторных отказов), то делается не два состояния $\mu\cdot\mathbf{o}_1$ и $\mu\cdot\mathbf{o}_2$, а одно, и не два перехода $\mu\text{—}\tau\rightarrow\mu\cdot\mathbf{o}_1$ и $\mu\text{—}\tau\rightarrow\mu\cdot\mathbf{o}_2$, а один переход. Для определённости можно всегда выбирать трассу без повторных отказов. Эта модификация корректна, поскольку трассы $\mu\cdot\mathbf{o}_1$ и $\mu\cdot\mathbf{o}_2$ имеют одинаковые продолжения. Для конечного алфавита из каждого состояния строящейся LTS будет вести только конечное число переходов, включая τ-переходы.

Заметим, что «компактное» преобразование $F2_{compact}L$ не алгоритмизуемо, поскольку основано на трассовых поддеревьях, которые могут быть бесконечными.

В целом вопросы разработки алгоритмов генерации тестов более тонкие, особенно для бесконечного алфавита. В частности, вместо преобразования из LTS в трассовую модель и обратно, эффективнее определять генерацию LTS-тестов непосредственно по LTS-спецификации. Здесь мы не будет углубляться в эти вопросы, поскольку они не входят в тему данной работы: нам было важно лишь показать возможность алгоритмизации при некоторых ограничениях на спецификацию.

4.8. Выводы

В этой главе мы определили LTS-модель как LTS с разрушением. Дивергенция как бесконечная цепочка τ-переходов допускается в LTS и моделируется Δ-действием при построении $\mathfrak{R}$-трасс LTS. Мы установили эквивалентность трассовой и LTS-моделей, что позволило развить LTS-теорию, опираясь на трассовую теорию. С точки зрения безопасности, конформности и генерации тестов LTS-модель более «избыточна», чем трассовая модель. Однако она имеет ряд преимуществ.

LTS-модель более наглядна и удобна для человеческого восприятия. Многие вещи можно увидеть в ней, так сказать, непосредственным созерцанием (когда пишут, что это «очевидно»). Трассовая модель не обладает такой наглядностью. В частности, генетические определения $\mathfrak{R}$- и F-трасс через LTS просты и понятны. Интенсиональные определения трассовых $\mathfrak{R}$- и F-моделей более сложные и далеко не очевидные. LTS более естественно «укладывается» в чёрный ящик машины тестирования, чем трассовая модель.

Но самым главным преимуществом LTS-модели является возможность определения композиции составной модели из моделей-компонентов, используя оператор параллельной композиции алгебры процессов (CCS в

данной работе). Соответственно, взаимодействие в LTS-теории выглядит как композиция LTS-реализации и LTS-окружения, а тестирование – как композиция LTS-реализации и LTS-теста. С этой точки зрения $\mathfrak{R}/\mathfrak{Q}$-машина тестирования интерпретируется как ограничение на LTS-окружение/тест, то есть на способ (правильного) взаимодействия с LTS-реализацией.

Таким образом, конформность и композиция определяются, фактически, на разных уровнях абстракции. Это становится причиной ряда проблем, главными из которых являются проблема рефлексивности спецификации и монотонности (сохранения конформности при композиции).

Новыми результатами являются:

1) Введение переходов по разрушению в LTS-модель.
2) Установление соответствия между моделью наблюдаемых трасс и LTS-моделью с учётом дивергенции и разрушения, в частности, с помощью «компактного» преобразования.
3) Определение гипотезы о безопасности и безопасной конформности для LTS.
4) Определение композиции LTS-моделей с учётом разрушения.
5) Метод генерации полного набора тестов, являющийся обобщением аналогичных методов для частных конформностей (в том числе, отношения ***ioco***) и дополнительно учитывающий разрушение и дивергенцию, то есть для безопасного тестирования. Метод алгоритмизуем при достаточно слабых ограничениях на LTS-спецификацию.

Глава 5. Пополнение спецификаций

Структура главы:

1. δ-семантика и отношение *ioco*
2. Допущения полноты в δ-семантике
3. Рефлексивность и транзитивность конформности
4. Существование пополнения
5. Проблема сохранения безопасности
6. Пополнение с не-отказами
7. Выводы

В Глава 2 мы говорили о двух проблемах, связанных с различием в определении безопасности для реализации (***safe in***) и спецификации (***safe by***). Это проблемы нерефлексивности и немонотонности конформности. В настоящей главе мы основное внимание уделим первой проблеме.

Отношение ***saco*** нерефлексивно только в том случае, когда есть ненаблюдаемые отказы (см. ниже Теорема 17:). Поэтому решением проблемы является переход от $\mathfrak{R}/\mathfrak{Q}$-семантики к $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантике. Такой переход делается с помощью преобразования спецификации, которое называется *пополнением* (как и сама преобразованная спецификация). При пополнении должна сохраняться конформность, то есть множество конформных реализаций.

Пополнение спецификации является также первым шагом к решению проблемы монотонности. Монотонность означает сохранение конформности при композиции LTS: композиция реализаций, конформных своим спецификациям, должна быть конформной композиции этих спецификаций. Эта проблема не может быть поставлена в теории $\mathfrak{R}$-трасс, поскольку там не определена композиция трассовых моделей. Проблема монотонности носит более общий характер, чем проблема пополнения, и остаётся даже в том случае,

когда нет Ω-кнопок. Решением проблемы является *монотонное* LTS-преобразование, сохраняющее конформность реализаций: композиция реализаций, конформных своим *преобразованным* спецификациям, конформна композиции этих *преобразованных* спецификаций. Композиция LTS-пополнения и монотонного LTS-преобразования является LTS-пополнением, которое можно назвать *монотонным пополнением*.

Сначала мы рассмотрим две эти проблемы на примере δ-семантики для реактивных систем и отношения ***ioco***, отличающегося от отношения $\boldsymbol{ioco}_\delta$ (***saco*** в δ-семантике) только более узким доменом реализаций и спецификаций. В этом случае любое пополнение оказывается монотонным при условии, что композиция пополненных спецификаций не выходит за пределы спецификационного домена отношения $\boldsymbol{ioco}_\delta$. Для отношения $\boldsymbol{ioco}_{\gamma\delta}$ (***saco*** в γδ-семантике), имеющего расширенный домен, это условие всегда выполнено.

После этого рассматривается преобразование пополнения в общем случае, которое определяется для трассовой модели. Сначала доказывается существование такого пополнения, когда в качестве преобразованной спецификации берётся объединение конформных реализаций. Далее изучается вопрос об изменении класса безопасных реализаций: он не должен расширяться. Для решения этой проблемы в спецификацию вводятся фиктивные действия – не-отказы. Даётся конструктивное (индуктивное) определение пополнения с не-отказами и кратко рассматриваются вопросы его алгоритмизации. Показывается, как выглядит такое пополнение для частных случаев δ-, γδ- и βγδ-семантик.

5.1. δ-семантика и отношение *ioco*

δ-семантика определяется для реактивных систем. Внешние действия делятся на стимулы и реакции: $L=?L\cup!L$, где $?L=\{?x \mid ?x\in L\}$ и $!L=\{!y \mid !y\in L\}$. Имеется одна $\mathfrak{R}$-кнопка для приёма всех реакций и соответствующий отказ, называемый стационарностью: $\mathfrak{R}=\{\delta\}=\{!L\}$. Для передачи каждого стимула имеется своя $\mathfrak{Q}$-кнопка: $\mathfrak{Q}=\{\{?x\} \mid ?x\in ?L\}$. От γδ-семантики δ-семантика отличается только запретом на разрушение и дивергенцию. Отношение ***ioco*** совпадает с отношением $\boldsymbol{ioco}_\delta$ (***saco*** в δ-семантике) на суженном домене реализаций и спецификаций: в реализации нет достижимых блокировок стимулов (реализация всегда принимает все стимулы – *input-enabledness*).

Определение 66: Пусть задан алфавит стимулов и реакций L.

- Обозначим множество ***F***-моделей без достижимых дивергенции и разрушения:

  $\boldsymbol{SMODEL}(L) =_{\text{def}} \boldsymbol{FMODEL}(L)\cap\mathcal{P}(L_{\mathcal{P}(L)}{}^{*})$.

- Обозначим множество ***F***-моделей без достижимых дивергенции, разрушения и блокировок стимулов:

  $\boldsymbol{IMODEL}(L) =_{\text{def}} \boldsymbol{FMODEL}(L)\cap\mathcal{P}(L_{\mathcal{P}(!L)}{}^{*})$.

- Определим отношение ***ioco*** для трассовых моделей как отношение ***saco*** в δ-семантике:

  $\mathbf{\Sigma}\in\boldsymbol{SMODEL}(L)$ & $\mathbf{I}\in\boldsymbol{IMODEL}(L)$:

  $\mathbf{I}$ ***ioco*** $\mathbf{\Sigma} =_{\text{def}} \mathbf{I}$ $\boldsymbol{ioco}_\delta$ $\mathbf{\Sigma} =_{\text{def}} \mathbf{I}$ $\boldsymbol{saco}_{\{\delta\}/\varnothing}$ $\mathbf{\Sigma}$.

- Обозначим класс строго-конвергентных LTS-моделей без достижимого разрушения:

  $\boldsymbol{SLTS}(L) =_{\text{def}} \{\mathbf{s}\in\boldsymbol{LTS}(L_\gamma) \mid \boldsymbol{F}(\mathbf{s})\in\boldsymbol{SMODEL}(L)\}$.

- Обозначим класс строго-конвергентных LTS-моделей без достижимых разрушения и блокировок стимулов:

  $\boldsymbol{ILTS}(\mathtt{L}) =_{\text{def}} \{\mathbf{S}\in\boldsymbol{LTS}(\mathtt{L}_\gamma) \mid \boldsymbol{F}(\mathbf{S})\in\boldsymbol{IMODEL}(\mathtt{L})\}.$

- Определим отношение ***ioco*** для LTS-моделей :

  $\mathbf{S}\in\boldsymbol{SLTS}(\mathtt{L})\ \&\ \mathbf{I}\in\boldsymbol{ILTS}(\mathtt{L})\ :\ \mathbf{I}\ \boldsymbol{ioco}\ \mathbf{S} =_{\text{def}} \boldsymbol{F}(\mathbf{I})\ \boldsymbol{ioco}\ \boldsymbol{F}(\mathbf{S}).$

□

Отношение ***ioco*** означает, что после общей δ-трассы реализации и спецификации любая реакция или стационарность, которая есть в реализации, должна быть и в спецификации. Гипотеза о безопасности, как предусловие тестирования, требует, чтобы каждый стимул принимался реализацией после любой трассы, то есть не было достижимых блокировок стимулов. Если в спецификации также нет достижимых блокировок стимулов, то отношение ***ioco*** сводится к вложенности δ-трасс. Это утверждение под названием *Lemma A.4* доказано в [67] для LTS-моделей. Поскольку мы доказали эквивалентность трассовой и LTS-моделей по отношению ***saco***, здесь мы даём только формулировку этого утверждения для трассовой и LTS-моделей в наших обозначениях и опускаем доказательство.

Утверждение: Пусть задан алфавит стимулов и реакций $\mathtt{L}$. Тогда:

$\forall\boldsymbol{\Sigma},\mathbf{I}\in\boldsymbol{SMODEL}(\mathtt{L})\ (\ \mathbf{I}\ \boldsymbol{ioco}\ \boldsymbol{\Sigma} \Leftrightarrow \mathbf{I}^{\downarrow}\mathtt{L}_{\{\delta\}} \subseteq \boldsymbol{\Sigma}^{\downarrow}\mathtt{L}_{\{\delta\}}\ )$ и

$\forall\mathbf{S},\mathbf{I}\in\boldsymbol{SLTS}(\mathtt{L})\ (\ \mathbf{I}\ \boldsymbol{ioco}\ \mathbf{S} \Leftrightarrow \boldsymbol{F}(\mathbf{I})^{\downarrow}\mathtt{L}_{\{\delta\}} \subseteq \boldsymbol{F}(\mathbf{S})^{\downarrow}\mathtt{L}_{\{\delta\}}\ ).$

□см.[67]

Из этого утверждения непосредственно следует рефлексивность и транзитивность отношения ***ioco*** для моделей без достижимых блокировок стимулов. Для таких моделей в [67] также доказана монотонность (*Theorem 9*).

Мы даём здесь только формулировку (без доказательства) этого утверждения для оператора композиции $\rceil\!\downarrow$.[23]

Утверждение: Пусть заданы алфавиты стимулов и реакций A и B. Тогда:

$\forall \mathbf{S}_1, \mathbf{I}_1 \in \boldsymbol{SLTS}(A) \quad \forall \mathbf{S}_2, \mathbf{I}_2 \in \boldsymbol{SLTS}(B)$

$\mathbf{I}_1 \ \boldsymbol{ioco} \ \mathbf{S}_1 \ \& \ \mathbf{I}_2 \ \boldsymbol{ioco} \ \mathbf{S}_2 \Rightarrow \mathbf{I}_1 \rceil\!\downarrow \mathbf{S}_1 \ \boldsymbol{ioco} \ \mathbf{I}_2 \rceil\!\downarrow \mathbf{S}_2$.

□см.[67]

В общем случае, когда в спецификации допускаются достижимые блокировки, отношение ***ioco*** нерефлексивно, транзитивно, и немонотонно.

Нерефлексивность. Любая спецификация, в которой достижимы блокировки стимулов, не конформна по ***ioco*** самой себе, поскольку не входит в домен реализаций отношения.

Заметим, что если δ-трассы спецификации не продолжаются одновременно стимулом и его блокировкой, то имеющиеся блокировки никак не проявляются в тестировании этой спецификации как реализации. Действительно, после δ-трассы, продолжаемой блокировкой стимула и не продолжаемой самим стимулом, стимул не посылается в реализацию при тестировании по ***ioco***, и реализация может как принимать, так и блокировать этот стимул. Иными словами, спецификации без совмещения после δ-трассы приёма стимула и его блокировки рефлексивны по отношению $\boldsymbol{ioco}_\delta$. Однако, если спецификация содержит δ-трассу, продолжаемую как стимулом, так и его блокировкой, то она не конформна самой себе по отношению $\boldsymbol{ioco}_\delta$, поскольку не является безопасной.

Транзитивность. Если $\mathbf{S}_1$ ***ioco*** $\mathbf{S}_2$ и $\mathbf{S}_2$ ***ioco*** $\mathbf{S}_3$, то в LTS $\mathbf{S}_1$ и $\mathbf{S}_2$ нет достижимых блокировок, поэтому $\boldsymbol{F}(\mathbf{S}_1)^{\downarrow}\mathrm{L}_{\{\delta\}} \subseteq \boldsymbol{F}(\mathbf{S}_2)^{\downarrow}\mathrm{L}_{\{\delta\}}$. Тогда, если δ-трасса σ, общая для $\mathbf{S}_1$ и $\mathbf{S}_3$, продолжается в $\mathbf{S}_1$ некоторой реакцией или

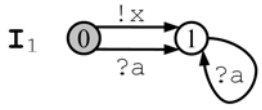
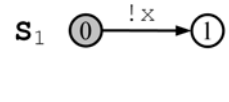
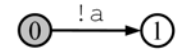
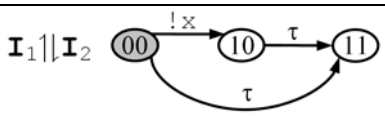
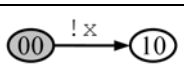
---

[23] В [67] используется оператор композиции $\rceil\!\|$ для алгебры процессов CSP и оператор сокрытия ***Hide***, превращающий синхронные переходы в τ-переходы.

стационарностью u, то это же имеет место и в $\mathbf{S}_2$. А тогда, поскольку $\mathbf{S}_2$ ***ioco*** $\mathbf{S}_3$, δ-трасса σ продолжается в $\mathbf{S}_3$ символом u, что и требуется для конформности $\mathbf{S}_1$ ***ioco*** $\mathbf{S}_3$.

Как мы покажем ниже (следствие из Леммы 15), транзитивность имеет место и для отношения $\boldsymbol{ioco}_\delta$ в том случае, когда блокировка стимула допускается в спецификации, но без совмещения с приёмом стимула после той же самой δ-трассы. В реализации в этом случае также может допускаться блокировка стимула после тех δ-трасс, которых нет в спецификации или которые продолжаются в спецификации этой блокировкой (и не продолжаются, тем самым, самим стимулом).

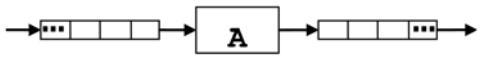
Немонотонность. Пример немонотонности приведён на Рис.45. Здесь в композиции конформных реализаций пустая трасса продолжается стационарностью δ, а в композиции спецификаций – только реакцией !x.

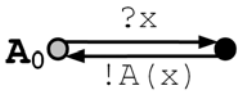
Пример немонотонности отношения ***ioco***			
алфавиты	реализации	спецификации	конформность
A={?a,!x}			$\mathbf{I}_1$ ***ioco*** $\mathbf{S}_1$
B={!a}			$\mathbf{I}_2$ ***ioco*** $\mathbf{S}_2$
A⫿B={!x}			$\mathbf{I}_1 \text{⫿} \mathbf{I}_2$ ~~***ioco***~~ $\mathbf{S}_1 \text{⫿} \mathbf{S}_2$
Рис.45.			

Другой пример – это система **A** на Рис.18, асинхронно тестируемая через две неограниченные очереди: входную и выходную (Рис.46). Через неограниченную входную очередь всегда можно послать два стимула x подряд: очередь их примет, а реализация потом будет выбирать стимулы из очереди. Если после этого принимать реакции, то, согласно спецификации A_0, должны быть последовательно приняты две реакции A(x). В то же время, хотя реализация A_1 конформна, при асинхронном тестировании в ней будет

обнаружена ошибка, поскольку второй реакции может не быть (стационарность в начальном состоянии).

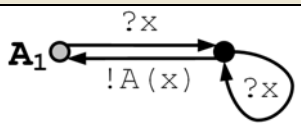
Пример несохранения отношения ***ioco*** при асинхронном тестировании





A_0 *saco* A_0

$A_0 \| Q$ *saco* $A_0 \| Q$



A_1 *saco* A_0

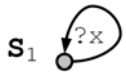
$A_1 \| Q$ ~~*saco*~~ $A_0 \| Q$

Рис.46.

Таким образом, если найти пополнение спецификаций в δ-семантике, избавляющее спецификацию от блокировок стимулов и сохраняющее множество конформных реализаций, то это решает обе проблемы: рефлексивности и монотонности.

Однако остаётся проблема выхода за пределы домена отношения ***ioco*** при композиции (как реализаций, так и спецификаций). Дело в том, что композиция строго-конвергентных LTS может иметь достижимую дивергенцию. Такая дивергенция возникает как бесконечная последовательность синхронных переходов. Пример приведён на Рис.47.

Пример дивергенции в композиции строго-конвергентных LTS



A={?x}

B={!x}

A||B=∅

Рис.47.

Поэтому предпочтительно рассматривать не δ-семантику, а γδ-семантику, и вместо отношения ***ioco*** использовать близкое к нему отношение $\boldsymbol{ioco}_{\gamma\delta}$, допускающее разрушение и дивергенцию. В [27] показано, что, если моделировать дивергенцию разрушением, то при отсутствии блокировок для монотонности достаточно однородности, которая означает: если в стабильном состоянии s, достижимом по безопасной δ-трассе, одна реакция разрушающая

$\exists(!y)\in L\ \exists s`\ s\text{—}!y\rightarrow s`=\langle\gamma\rangle\Rightarrow$, то и все реакции разрушающие $\forall!t\in L\ \exists s``\ s\text{—}!t\rightarrow s``=\langle\gamma\rangle\Rightarrow$. Понятно, что любую спецификацию, в том числе пополненную, легко можно сделать однородной, добавив нужные переходы по реакциям $s\text{—}!t\rightarrow s`$.

5.2. Допущения полноты в δ-семантике

Структура раздела:

1. Виды пополнения состояний
2. Несохранение *ioco* при пополнении состояний
3. Демоническое и гамма-пополнения состояний и трасс

В δ-семантике блокировки стимулов не наблюдаемы Из-за этого наличие блокировок в спецификации приводит к проблемам нерефлексивности и немонотонности. Поэтому отсутствие приёма стимула в том или ином состоянии спецификации иногда интерпретируется как недоспецификация, то есть как явно не специфицированное поведение при передаче этого стимула. При этом предполагается, что для такого, явно не специфицированного, поведения существует некоторая трактовка, определяющая «умалчиваемое поведение». Такая трактовка называется допущением полноты (completeness assumption), она определяет соответствующее пополнение спецификации. Таких пополнений предложено довольно много. Обзор их различных видов или наиболее важные из них можно найти в [27,66,71,113]. Заметим, что то или иное допущение полноты и основанное на нём пополнение делаются ещё до того, как встаёт вопрос о конформности. Пополнение, основанное на допущении полноты, добавляет в LTS-модель переходы по стимулам из состояний, из которых такие переходы не выходили. Поэтому такое пополнение можно назвать пополнением состояний.

5.2.1. Виды пополнения состояний

Рассмотрим несколько видов пополнения состояний (Рис.48).

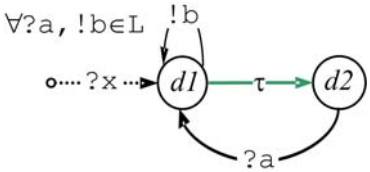
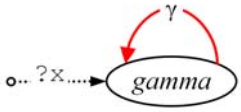
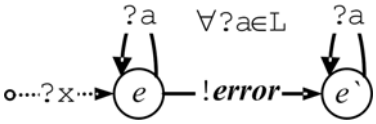
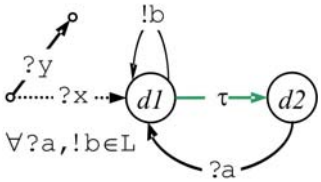
Виды пополнения состояний		
Ангельское пополнение	Демоническое пополнение	Гамма-пополнение
		
Реакция об ошибке	Демоническое пополнение в рецептивных состояниях	Гамма-пополнение в стационарных состояниях
		

Рис.48.

Каждый вид пополнения представлен ниже в двух вариантах в зависимости от того, идёт ли речь об отсутствии перехода по стимулу в любом состоянии или только в стабильном, где такое отсутствие перехода означает блокировку стимула. Поскольку, по крайней мере, до пополнения подразумевается δ-семантика, в которой нет разрушения, стабильность означает отсутствие только τ-переходов. Условие стабильности заключено в квадратные скобки “[]”.

Ангельское пополнение (*angelic completion*) [66,67,113].

Для неспецифицированного стимула определяется переход-петля по этому стимулу: стимул принимается, но игнорируется, состояние не меняется:

$[s\text{—}\tau\!\not\to] \;\&\; s\text{—}?x\!\not\to \;\vdash\; s\text{—}?x\!\to\! s.$

Демоническое пополнение (*demonic completion*) [66,67,128].

Для неспецифицированного стимула определяется переход по этому стимулу с произвольным дальнейшим поведением без блокировок и разрушения: $\forall ?a, !b \in L$

$[s\text{—}\tau\not\to] \;\&\; s\text{—}?x\not\to \;\vdash\; s\text{—}?x\to d1\text{—}!b\to d1\text{—}\tau\to d2\text{—}?a\to d1.$

Реакция об ошибке [113].

Для неспецифицированного стимула определяется переход в специальное состояние с последующей выдачей реакции об ошибке: $\forall ?a\in L$

$[s\text{—}\tau\not\to] \;\&\; s\text{—}?x\not\to \;\vdash\; s\text{—}?x\to e\text{—}?a\to e\text{—}!\boldsymbol{error}\to e`\text{—}?a\to e`.$

Это пополнение можно трактовать двояко:

1) Реакция об ошибке $!\boldsymbol{error}$ добавляется в алфавит реакций реализации. Реализация должна после приёма стимула $?x$ выдавать такую реакцию и не должна выдавать другую реакцию или отказ δ.

2) Реакция об ошибке $!\boldsymbol{error}$ *не* добавляется в алфавит реакций реализации. Любая реакция или отказ δ в реализации после приёма стимула $?x$ считается ошибочной. Это означает, что конформная реализация не должна проходить при тестировании трассу, после которой в спецификации (до пополнения) стимул $?x$ не специфицирован (ни в каком состоянии).

Гамма-пополнение [22,23,25,27].

Для неспецифицированного стимула определяется переход по этому стимулу, ведущий к разрушению:

$[s\text{—}\tau\not\to] \;\&\; s\text{—}?x\not\to \;\vdash\; s\text{—}?x\to gamma\text{—}\gamma\to gamma.$

Возможны комбинации этих вариантов в зависимости от состояния LTS.

Демоническое пополнение в рецептивных состояниях.

В [103] предлагается принимать либо все стимулы, либо ни одного. Различаются состояния, в которых есть специфицированные стимулы, и состояния, в которых таких стимулов нет. Состояния первого вида можно 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 (демоническим) поведением. В нерецептивных состояниях приём стимулов вообще не ведётся.

Если нерецептивное состояние стабильно, то все стимулы блокируются: $\forall ?a,!b \in L$

$\exists ?c\ s \text{—} ?c \rightarrow\ \&\ s \text{—} ?x \not\rightarrow \vdash s \text{—} ?x \rightarrow d1 \text{—} !y \rightarrow d1 \text{—} \tau \rightarrow d2 \text{—} ?a \rightarrow d1.$

Гамма-пополнение в стационарных состояниях.

В работах ИСП РАН [10,13] различаются стационарные (нет внутренней активности и выдачи реакций) и нестационарные (есть внутренняя активность или выдача реакций) состояния. В стационарном состоянии неспецифицированный стимул принимается и ведёт к разрушению, а в нестационарном – не принимается. Если нестационарное состояние стабильно, неспецифицированный стимул блокируется.

$\forall !c\ s \text{—} !c \not\rightarrow\ \&\ s \text{—} \tau\gamma \not\rightarrow\ \&\ s \text{—} ?x \not\rightarrow \vdash s \text{—} ?x \rightarrow gamma \text{—} \gamma \rightarrow gamma.$

Основанием для такого пополнения является предположение, что реактивная 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. Если для принятого стимула у неё не определено поведение, это ведёт (более строго, может вести) к разрушению системы. Это можно понимать как нарушение предусловия операции (стимула). Поскольку мы должны избегать разрушения, тест должен выдавать только такой стимул, который допустим во всех стационарных состояниях в конце пройденной βγδ-трассы.

5.2.2. Несохранение *ioco* при пополнении состояний

Все рассмотренные выше пополнения состояний не сохраняют отношение ***ioco***: множество реализаций, ***ioco***-конформных исходной спецификации, не совпадает с множеством реализаций, ***ioco***-конформных пополненной спецификации. Можно выделить два случая интерпретации блокировки {?x}

после δ-трассы μ в зависимости от того, A) продолжается δ-трасса μ самим стимулом `?x` или B) нет.

A) Пусть δ-трасса μ продолжается как стимулом `?x`, так и его блокировкой `{?x}`. Это значит, что δ-трасса заканчивается, по крайней мере, в двух состояниях, в одном из которых есть переход по стимулу `?x`, а в другом нет.

Все рассмотренные выше пополнения состояний, кроме одного случая, ослабляют конформность (Рис.49).

Ангельское пополнение разрешает после δ-трассы $\mu\cdot\langle$`?x`$\rangle$ то поведение, которое в исходной спецификации было после δ-трассы $\mu\cdot\langle$`{?x}`$\rangle$. Однако это могло не разрешаться исходной спецификацией.

Демоническое пополнение разрешает любое поведение реализации после δ-трассы $\mu\cdot\langle$`?x`$\rangle$. Однако в исходной спецификации могло разрешаться не любое поведение. Для демонического пополнения в рецептивных состояниях то же самое возможно, когда δ-трасса $\mu\cdot\langle$`{?x}`$\rangle$ заканчивается хотя бы в одном рецептивном состоянии.

Пополнение «реакция об ошибке» имеет две трактовки.

1) Реакция об ошибке `!`***error*** добавляется в алфавит стимулов и реакций реализации. В этом случае после трассы $\mu\cdot\langle$`?x`$\rangle$ разрешается реакция $\langle$`!`***error***$\rangle$, которая до пополнения не разрешалась.

2) Реакция об ошибке `!`***error*** *не* добавляется в алфавит стимулов и реакций реализации. В этом случае конформная реализация не может выдать реакцию `!`***error***, и *нет ослабления* соответствия.

После гамма-пополнения не проверяется поведение реализации после δ-трассы $\mu\cdot\langle$`?x`$\rangle$, хотя в исходной спецификации могло разрешаться не любое такое поведение. Для гамма-пополнения в стационарных состояниях то же

самое может иметь место, когда δ-трасса μ заканчивается хотя бы в одном стационарном состоянии.

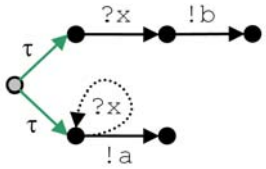
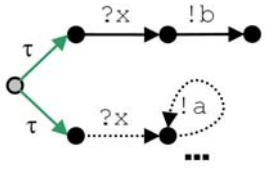
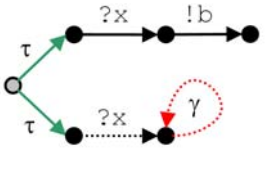
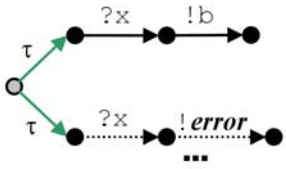
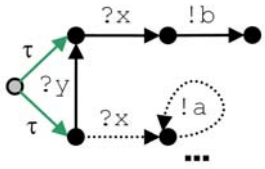
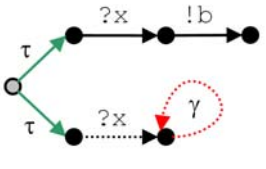
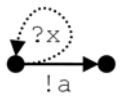
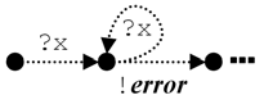
Ослабление конформности при пополнении состояний		
Ангельское пополнение	Демоническое пополнение	Гамма-пополнение
		
Реакция об ошибке (1)	Демоническое пополнение в рецептивных состояниях	Гамма-пополнение в стационарных состояниях
		
Для «Реакция об ошибке (1)» `L={?x,?y,!a,!b,!`***error***`}` В остальных случаях `L={?x,?y,!a,!b}` При пополнении после пустой трассы разрешается реакция `!a` (или `!`***error***), которая до пополнения не разрешалась.		

Рис.49.

В) Теперь пусть δ-трасса μ не продолжается стимулом `?x`. Прежде всего, отметим, что исходная спецификация разрешала любое поведение после δ-трассы $\mu\cdot\langle$`?x`$\rangle$, поскольку при тестировании стимул не посылался в реализацию и, тем самым, не проверялось, какое поведение имеет реализация после приёма стимула. Здесь предполагается, что в δ-семантике реализация полностью определена по стимулам (*input enabled*) и в ней нет разрушения.

Все рассмотренные выше пополнения состояний, кроме двух (и комбинированных вариантов), усиливают соответствие (Рис.50).

Усиление конформности при пополнении состояний		
Ангельское пополнение	Реакция об ошибке (1)	Реакция об ошибке (2)
		
Для «Реакция об ошибке (1)» L={?x,!a,!b,!***error***} В остальных случаях L={?x,!a,!b} При пополнении после трассы ⟨?x⟩ не разрешается реакция !b, которая до пополнения разрешалась.		
Рис.50.		

Ангельское пополнение усиливает соответствие: после δ-трассы $\mu\cdot\langle ?x\rangle$ разрешается только то поведение, которое в исходной спецификации было после δ-трассы μ, а оно могло быть не любым.

Демоническое пополнение *не усиливает* соответствие, поскольку разрешает любое поведение реализации после приёма стимула. То же относится к демоническому пополнению в рецептивных состояниях.

Пополнение «реакция об ошибке» усиливает соответствие: после δ-трассы $\mu\cdot\langle ?x\rangle$ разрешает единственную реакцию !***error***.

Гамма-пополнение *не усиливает* соответствие, поскольку по-прежнему не посылается стимул ?x после δ-трассы μ, но только теперь по причине того, что он разрушающий после этой δ-трассы и его запрещено посылать в реализацию при тестировании. То же относится к гамма-пополнению в стационарных состояниях.

5.2.3. Демоническое и гамма-пополнения состояний и трасс

Таким образом, единственные пополнения состояний, которые *не усиливают* конформность, – это демоническое и гамма-пополнения. Сравним эти два вида пополнения состояний.

Демоническое пополнение эквивалентно использованию в конформности ***ioco*** не всех δ-трасс, а только их подмножества ***Utraces*** [66]. Это такие δ-

трассы, в которых каждый префикс, продолжаемый в δ-трассе стимулом, не имеет в LTS-спецификации $\mathbf{s}\in\boldsymbol{LTS}(\mathtt{L})$ продолжения блокировкой этого стимула:

$$\boldsymbol{Utraces}(\mathbf{s}) =_{\text{def}} \{\sigma\in\boldsymbol{F}(\mathbf{s})^{\downarrow\mathtt{L}_\delta} \mid \forall ?x,\mu\ (\mu\cdot\langle ?x\rangle\le\sigma \Rightarrow \mu\cdot\langle\{?x\}\rangle\notin\boldsymbol{F}(\mathbf{s}))\}.$$

В свою очередь, это эквивалентно использованию в конформности ***ioco*** δ-трасс, безопасных после гамма-пополнения состояний спецификации, поскольку, очевидно, $\boldsymbol{Utraces}(\mathbf{s})=\boldsymbol{SafeBy}\circ\boldsymbol{F}\circ\boldsymbol{Гамма\text{-}пополнение}(\mathbf{s})$. Гамма-пополнение и использование затем для тестирования его безопасных δ-трасс удобнее 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 [66]. Это даёт возможность избегать лишних тестов при генерации полного тестового набора по пополненной спецификации, в частности, при переходе от синхронного к асинхронному тестированию.

Хотя в общем случае при гамма-пополнении возможно ослабление конформности, для примера на Рис.46 этого не происходит (Рис.51). Второй стимул оказывается опасным в спецификации, и поэтому он не посылается как при синхронном, так и при асинхронном тестировании.

Пример гамма-пополнения

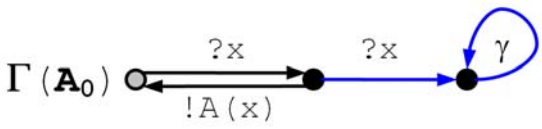
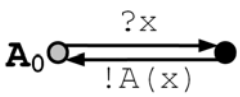
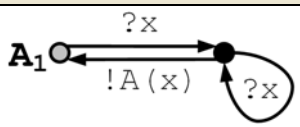
Гамма-пополнение спецификации $\mathbf{A}_0$		
	 $\mathbf{A}_0$ *saco* $\Gamma(\mathbf{A}_0)$ $\mathbf{A}_0\!\upharpoonright\!\mathbf{Q}$ *saco* $\Gamma(\mathbf{A}_0)\!\upharpoonright\!\mathbf{Q}$	 $\mathbf{A}_1$ *saco* $\Gamma(\mathbf{A}_0)$ $\mathbf{A}_1\!\upharpoonright\!\mathbf{Q}$ *saco* $\Gamma(\mathbf{A}_0)\!\upharpoonright\!\mathbf{Q}$

Рис.51.

Причина ослабления конформности при демоническом или гамма-пополнении в том, что δ-трасса μ может заканчиваться в нескольких

состояниях, в одном из которых она продолжается стимулом ?x, а в другом, состоянии s, – его блокировкой {?x}. Отношение ***ioco***, фактически, требует игнорировать блокировку {?x}. Однако для этого мы должны определить переход s—?x→. Но какое поведение нужно определить после добавленного перехода? Чтобы не ослаблять конформность, такое поведение должно быть пересечением поведений после всех δ-трасс $\sigma\cdot\langle ?x\rangle$, где $s\in(\mathbf{s}\ \textit{after}\ \sigma)$. Это должно быть множество δ-трасс

$$\mathbf{T}(s) =_{def} \cap(\{\boldsymbol{F}(\mathbf{s})^{\downarrow L_\delta}\ \textit{after}\ \sigma \mid s\in(\mathbf{s}\ \textit{after}\ \sigma)\ \&\ \sigma\cdot\langle ?x\rangle\in\boldsymbol{F}(\mathbf{s})\}).$$

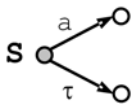
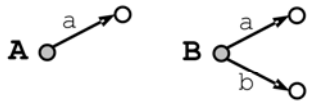
Однако здесь возникают две проблемы. Во-первых, в состоянии s может заканчиваться δ-трасса, которая не продолжается стимулом ?x. Для того, чтобы не произошло усиления конформности, множество $\mathbf{T}(s)$ должно быть «демоническим», то есть содержать все согласованные δ-трассы, а это вовсе не обязательно. Во-вторых, множество $\mathbf{T}(s)$ должно быть δ-моделью. Однако, хотя каждое множество $\boldsymbol{F}(\mathbf{s})^{\downarrow L_\delta}\ \textit{after}\ \sigma$ является δ-моделью (Лемма 2) , их пересечение вовсе не обязательно будет δ-моделью (Теорема 3: и п.3.3.2). Это видно в примере на Рис.15.

Общий вывод, который можно сделать из этого анализа пополнения состояний заключается в том, что нам требуется не пополнение состояний, а пополнение трасс. Любое пополнение состояний, вообще говоря, не сохраняет конформность. Такой вывод был сделан в [105], правда не для отношения ***ioco*** (и, тем более, не для отношения ***saco*** в произвольной $\mathfrak{R}/\mathfrak{Q}$-семантике). С другой стороны, такое пополнение трасс можно выполнить как демоническое или гамма-пополнение состояний *преобразованной* LTS-спецификации. Преобразованная LTS должна быть эквивалентна исходной по множеству конформных реализаций, но не содержать трасс, имеющих смешанное продолжение стимулом и его блокировкой. В преобразованной LTS в любом стабильном состоянии, в котором нет перехода по стимулу ?x, должны

заканчиваться только те δ-трассы, которые не продолжаются стимулом ?x. Поэтому демоническое или гамма-пополнение, не усиливающие конформность в общем случае, не ослабят её для преобразованной спецификации, но позволят избавиться от оставшихся блокировок. Более подробно о проблемах пополнения спецификаций в δ- и γδ-семантиках говорится в [27]. Здесь же мы вернёмся к общему случаю произвольной $\mathfrak{R}/\mathfrak{Q}$-семантики, для которой и определим пополнение, сохраняющее конформность. Соответствующее пополнение для δ- и γδ-семантик будет частным случаем общего пополнения.

5.3. Рефлексивность и транзитивность конформности

Отношение ***saco***, вообще говоря, нерефлексивно и нетранзитивно. Примеры $\mathfrak{R}/\mathfrak{Q}$-семантик и отношений ***safe by*** приведены на Рис.52.

<table>
<tr><th colspan="2">Нерефлексивность отношения saco</th></tr>
<tr><td>L={a} $\mathfrak{R}$=∅ $\mathfrak{Q}$={{a}} LTS =  </td><td>S <s>safe for</s> S
{a} safe by F(S) after ϵ
{a} <s>safe in</s> F(S) after ϵ</td></tr>
<tr><th colspan="2">Нетранзитивность отношения saco</th></tr>
<tr><td>L={a,b,c} $\mathfrak{R}$=∅ $\mathfrak{Q}$={{a,b},{a,c},{b,c}}
LTS = 
{a,b} safe by F(B) after ϵ {b,c} safe by F(C) after ϵ
Другие кнопки опасны после ϵ в B и C</td><td>A saco B
B saco C
A <s>safe for</s> C
{b,c} safe in F(B) after ϵ
{b,c} <s>safe by</s> F(B) after ϵ</td></tr>
<tr><td colspan="2">Рис.52.</td></tr>
</table>

Причина нерефлексивности и нетранзитивности в различии отношений ***safe by*** и ***safe in*** (на Рис.52 выделено серым фоном). Определим достаточные условия рефлексивности и транзитивности, связывающие эти два отношения безопасности. Будем считать, по-прежнему, что в алфавите L⊆Z задана $\mathfrak{R}/\mathfrak{Q}$-семантика.

Определение 67: Пусть задана ***F***-спецификация Σ. Определим два условия на спецификацию:

- *Условие рефлексивности*: безопасность кнопки по ***safe by*** после трассы, безопасной по обоим отношениям, влечёт её безопасность по ***safe in*** после этой же трассы: $\forall \sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma) \quad \forall P \in \mathfrak{R} \cup \mathfrak{Q}$

  $(P$ ***safe by*** Σ ***after*** $\sigma \Rightarrow P$ ***safe in*** Σ ***after*** $\sigma)$.
- *Условие транзитивности*: безопасность кнопки по ***safe in*** после трассы, безопасной по обоим отношениям, влечёт её безопасность по ***safe by*** после этой же трассы: $\forall \sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma) \quad \forall P \in \mathfrak{R} \cup \mathfrak{Q}$

  $(P$ ***safe in*** Σ ***after*** $\sigma \Rightarrow P$ ***safe by*** Σ ***after*** $\sigma)$.

□

Лемма 14: Пусть задана ***F***-спецификация Σ.

- Рефлексивность: Если для Σ выполнено условие рефлексивности, то $\mathit{SafeBy}(\Sigma) \subseteq \mathit{SafeIn}(\Sigma)$.
- Транзитивность: Если для Σ выполнено условие транзитивности, то $\mathit{SafeBy}(\Sigma) \supseteq \mathit{SafeIn}(\Sigma)$.

□469

Безопасность $\mathfrak{R}$-кнопок одинакова по обоим отношениям ***safe by*** и ***safe in***. Разрушающая $\mathfrak{Q}$-кнопка опасна по обоим отношениям. Неразрушающая $\mathfrak{Q}$-кнопка “Q” безопасна по ***safe in***, если она не вызывает ненаблюдаемый отказ Q.

Поэтому условие рефлексивности эквивалентно тому, что безопасность $\mathfrak{Q}$-кнопки “Q” по ***safe by*** после трассы, безопасной по обоим отношениям, влечёт отсутствие продолжения этой трассы отказом Q:

$\forall \sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma) \quad \forall Q \in \mathfrak{Q}$

$(Q$ ***safe by*** Σ ***after*** $\sigma \Rightarrow \sigma \cdot \langle Q \rangle \notin \Sigma)$.

Поскольку из условия рефлексивности следует $\mathit{SafeBy}(\Sigma) \subseteq \mathit{SafeIn}(\Sigma)$, это условие можно записать также в эквивалентной форме:

$\forall \sigma \in \mathit{SafeBy}(\Sigma) \quad \forall Q \in \mathfrak{Q}$

$(Q \ \mathit{safe\ by}\ \Sigma\ \mathit{after}\ \sigma \Rightarrow \sigma \cdot \langle Q \rangle \notin \Sigma)$.

Аналогично, условие транзитивности эквивалентно тому, что отсутствие продолжения трассы, безопасной по обоим отношениям, $\mathfrak{Q}$-отказом Q, соответствующим неразрушающей кнопке “Q”, влечёт безопасность кнопки “Q” по *safe by* после этой трассы:

$\forall \sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma) \quad \forall Q \in \mathfrak{Q}$

$(\forall z \in Q \ \ \sigma \cdot \langle z, \gamma \rangle \notin \Sigma \ \ \& \ \ \sigma \cdot \langle Q \rangle \notin \Sigma \Rightarrow Q \ \mathit{safe\ by}\ \Sigma\ \mathit{after}\ \sigma)$.

Поскольку из условия транзитивности следует $\mathit{SafeBy}(\Sigma) \supseteq \mathit{SafeIn}(\Sigma)$, это условие можно записать также в эквивалентной форме:

$\forall \sigma \in \mathit{SafeIn}(\Sigma) \quad \forall Q \in \mathfrak{Q}$

$(\forall z \in Q \ \ \sigma \cdot \langle z, \gamma \rangle \notin \Sigma \ \ \& \ \ \sigma \cdot \langle Q \rangle \notin \Sigma \Rightarrow Q \ \mathit{safe\ by}\ \Sigma\ \mathit{after}\ \sigma)$.

Лемма 15: Пусть задана ***F***-спецификация Σ.

- Рефлексивность: Если для спецификации Σ выполнено условие рефлексивности, то Σ *saco* Σ.
- Транзитивность: Если для спецификации Σ выполнено условие транзитивности, то для любой ***F***-реализации I и ***F***-спецификации Ω со своим отношением *safe by* из I *saco* Σ и Σ *saco* Ω следует I *saco* Ω.

□469

Условие рефлексивности является также необходимым условием. Действительно, пусть Σ *saco* Σ. Тогда Σ *safe for* Σ, что влечёт

$\forall \sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma) \quad \forall P \in \mathfrak{R} \cup \mathfrak{Q}$

$(P \ \mathit{safe\ by}\ \Sigma\ \mathit{after}\ \sigma \Rightarrow P \ \mathit{safe\ in}\ \Sigma\ \mathit{after}\ \sigma)$,

то есть выполнение условия рефлексивности.

В то же время условие транзитивности не является необходимым. Пример приведён на Рис.53.

Условие транзитивности не необходимо	
LTS **S**  $\Sigma=F(\mathbf{S})$ $\mathfrak{R}=\varnothing$, $\mathfrak{Q}=\{\{a\},\{a,b\}\}$	$\forall\sigma\in\{a\}^*$ $\{a,b\}$ ***safe by*** Σ ***after*** σ $\{a\}$ ~~***safe by***~~ Σ ***after*** σ $\{a\}$ ***safe in*** Σ ***after*** σ **I** ***saco*** $\Sigma \Leftrightarrow$ **I** $=\Sigma$ Поэтому **I** ***saco*** Σ & Σ ***saco*** $\Omega \Rightarrow$ **I** ***saco*** Ω.
Рис.53.	

Следствие из Леммы 15: отношение $\boldsymbol{ioco}_\delta$ (***saco*** в δ-семантике) рефлексивно и транзитивно на классе спецификаций, в которых нет совмещения продолжения δ-трассы стимулом и его блокировкой, поскольку в этом случае отношения ***safe by*** и ***safe in*** совпадают.

При отсутствии $\mathfrak{Q}$-кнопок отношения ***safe by*** и ***safe in*** также совпадают. Как следствие, совпадают понятия безопасности трасс по этим отношениям ***SafeBy***=***SafeIn***.

Теорема 17: Если $\mathfrak{Q}$-кнопок нет, то отношение ***saco*** рефлексивно и транзитивно, то есть является предпорядком.

□472

Для дальнейшего нам понадобится также следующее утверждение.

Теорема 18: Если $\mathfrak{Q}$-кнопок нет, две ***F***-модели, конформные друг другу, имеют одинаковые множества безопасных $\mathfrak{R}$-трасс. Кроме того, они имеют одинаковые классы как конформных, так и безопасных реализаций.

□473

Заметим, что утверждение, обратное первому утверждению Теоремы, вообще говоря, не верно: две ***F***-модели могут иметь одинаковые безопасные $\mathfrak{R}$-трассы, но не быть конформными друг другу. Это показывает пример на Рис.33.

Также ниже мы увидим, что при наличии $\mathfrak{Q}$-кнопок классы безопасных реализаций у двух конформных друг другу спецификаций могут не совпадать.

Итак, для того, чтобы получить рефлексивную (и транзитивную) конформность, нужно перейти от спецификации Σ с отношением ***safe by*** Σ в $\mathfrak{R}/\mathfrak{Q}$-семантике к спецификации $\Sigma`$ в $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантике, сохраняя множество конформных реализаций. Мы покажем, что можно выбрать такую ***F***-модель $\Sigma`$ и такое отношение ***safe by*** $\Sigma`$, чтобы множества её конформных реализаций в $\mathfrak{R}/\mathfrak{Q}$- и $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантиках совпадали.

Определение 68: *Пополнением* ***F***-спецификации Σ будем называть такую ***F***-спецификацию $\Sigma`$, что имеет место: $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)=\mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\Sigma`)=\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma`)$.

Преобразование $\Sigma\to\Sigma`$ будем называть преобразованием пополнения или просто пополнением.

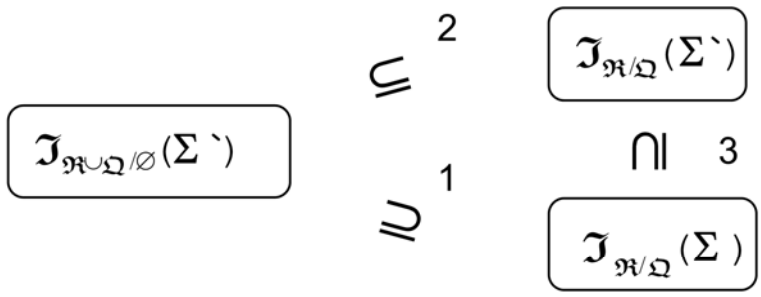
□

5.4. Существование пополнения

Во всём этом разделе мы будем предполагать, что в алфавите $L\subseteq Z$ задана $\mathfrak{R}/\mathfrak{Q}$-семантика и ***F***-спецификация $\Sigma\in$ ***FMODEL***(L) с отношением ***safe***$_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** Σ. Обозначим объединение реализаций, конформных спецификации Σ, через $\Sigma`=\cup\circ\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)$.

Прежде всего, отметим, что, по Теорема 4:, $\Sigma`$ является ***F***-моделью. Мы покажем, что при подходящем определении отношения ***safe***$_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\Sigma`$ ***F***-спецификация $\Sigma`$ будет пополнением ***F***-спецификации Σ. План доказательства изображён на Рис.54.

План доказательства эквивалентности спецификации и объединения конформных реализаций



Σ – исходная спецификация

$\Sigma` = \cup \circ \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)$ – объединение конформных реализаций

Рис.54.

Сначала покажем вложенность «1» (Рис.54).

Лемма 16: В $\mathfrak{R}/\varnothing$-семантике вложенность ***F***-реализации в ***F***-спецификацию влечёт конформность: $\mathbf{I} \subseteq \mathbf{\Sigma} \Rightarrow \mathbf{I}$ ***saco*** $\mathbf{\Sigma}$.

□474

Заметим, что в $\mathfrak{R}/\mathfrak{Q}$-семантике это утверждение, вообще говоря, не верно. Это следует, например, из нерефлексивности отношения ***saco*** при наличии $\mathfrak{Q}$-кнопок.

Вложенность «1» является простым следствием Леммы 16.

Лемма 17: $\mathbf{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma}) \subseteq \mathbf{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\mathbf{\Sigma`})$.

□475

Для дальнейшего нам понадобится утверждение о том, что объединение конформных реализаций также является конформной реализацией.

Лемма 18: $\mathbf{\Sigma`}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$.

□475

Теперь нам нужно определить отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\mathbf{\Sigma`}$. Определим его для безопасных трасс $\sigma \in \boldsymbol{SafeIn}(\mathbf{\Sigma`})$ как $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\mathbf{\Sigma`}$ ***after*** $\sigma$ = ***safe in*** $\mathbf{\Sigma`}$ ***after*** σ. Как отмечено в 3.6.2, отношение

safe*$_{\mathfrak{R}/\mathfrak{Q}}$ *by достаточно определить только на безопасных трассах, то есть будет ***SafeBy***$(\Sigma`)$=***SafeIn***$(\Sigma`)$. Тем не менее формально для опасных трасс $\sigma\notin$***SafeIn***$(\Sigma`)$ отношение ***safe*$_{\mathfrak{R}/\mathfrak{Q}}$ *by*** $\Sigma`$ ***after*** σ можно определить любым способом, удовлетворяющим трём правилам протокола. Такой способ всегда существует (3.6.2). Нам нужно показать, что отношение ***safe*$_{\mathfrak{R}/\mathfrak{Q}}$ *by*** на безопасных (по ***safe in***) трассах также удовлетворяет трём правилам протокола.

Сначала докажем два вспомогательных утверждения о свойствах отношения ***safe in*** в любой модели и в объединении конформных реализаций.

Лемма 19: Отношение ***safe in*** обладает следующим свойством: если кнопка опасна после $\mathfrak{R}$-трассы μ, то она опасна после любой $\mathfrak{R}$-трассы, получаемой из μ с помощью ***di***-операций:

$\forall \mathbf{I}\in$***FMODEL***$(\mathtt{L})\ \ \forall\mu\in\mathbf{I}^{\downarrow}\mathtt{L}_{\mathfrak{R}}\ \ \forall\mu`\in$***di***$(\mu)\ \ \forall \mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}$

$\mathtt{P}$ ~~***safe in***~~ $\mathbf{I}$ ***after*** $\mu\ \Rightarrow\ \mathtt{P}$ ~~***safe in***~~ $\mathbf{I}$ ***after*** $\mu`$.

□476

Определение 69: Пусть задана ***F***-модель Σ и множество пар (трасса, внешнее действие) $\mathbf{M}\subseteq\Sigma\times\mathtt{L}$ такое, что для каждой пары $(\mu,\mathtt{z})\in\mathbf{M}$ трасса $\mu\cdot\langle\mathtt{z}\rangle$ допустима и согласована. Обозначим множество трасс модели, пополненное согласованными продолжениями всех трасс из ***di***(μ) действием $\mathtt{z}$ и далее разрушением или дивергенцией для каждой пары $(\mu,\mathtt{z})\in\mathbf{M}$:

$\Sigma(\mathbf{M},\gamma)\ =_{\text{def}}\ \Sigma\cup\cup\{\boldsymbol{c}(\mu,\mathtt{z},\gamma)\mid(\mu,\mathtt{z})\in\mathbf{M}\}$,

$\Sigma(\mathbf{M},\Delta)\ =_{\text{def}}\ \Sigma\cup\cup\{\boldsymbol{c}(\mu,\mathtt{z},\Delta)\mid(\mu,\mathtt{z})\in\mathbf{M}\}$, где

$\boldsymbol{c}(\mu,\mathtt{z},\gamma)=\{\mu`\in$***di***$(\mu)\cdot\{\langle\mathtt{z}\rangle,\langle\mathtt{z},\gamma\rangle\}\mid\mu`\cdot\langle\mathtt{z}\rangle$ допустима и согласована$\}$,

$\boldsymbol{c}(\mu,\mathtt{z},\Delta)=\{\mu`\in$***di***$(\mu)\cdot\{\langle\mathtt{z}\rangle,\langle\mathtt{z},\Delta\rangle\}\mid\mu`\cdot\langle\mathtt{z}\rangle$ допустима и согласована$\}$.

□

Лемма 20: Пусть задана ***F***-модель $\mathbf{\Sigma}$ и множество пар (трасса, внешнее действие) $\mathbf{M}\subseteq\mathbf{\Sigma}\times L$ такое, что для каждой пары $(\mu,z)\in\mathbf{M}$ трасса $\mu\in\mathbf{\Sigma}$, а трасса $\mu\cdot\langle z\rangle$ допустима и согласована. Тогда $\mathbf{\Sigma}(\mathbf{M},\gamma),\mathbf{\Sigma}(\mathbf{M},\Delta)\in\boldsymbol{FMODEL}(L)$ и для каждой пары $(\mu,z)\in\mathbf{M}$ имеет место $\mu\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}(\mathbf{M},\gamma)$ и $\mu\cdot\langle z,\Delta\rangle\in\mathbf{\Sigma}(\mathbf{M},\Delta)$.

□476

Лемма 21: Если в $\mathbf{\Sigma`}$ после $\mathfrak{R}$-трассы действие не запрещено постфиксом отказов трассы и опасно (по ***safe in***), то оно разрушающее: $\forall\mu\in\mathbf{\Sigma`}^{\downarrow L_{\mathfrak{R}}}\ \forall z\in L\ z\notin\cup\circ\boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ & z ***safe in*** $\mathbf{\Sigma`}$ ***after*** $\mu\Rightarrow\mu\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma`}$.

□478

Лемма 22: На безопасных трассах $\sigma\in\boldsymbol{SafeIn}(\mathbf{\Sigma`})$ отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\mathbf{\Sigma`}$ ***after*** σ = ***safe in*** $\mathbf{\Sigma`}$ ***after*** σ удовлетворяет всем трём правилам протокола взаимодействия.

□479

Такое отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\mathbf{\Sigma`}$, очевидно, удовлетворяет условиям рефлексивности и транзитивности.Из условия рефлексивности следует вложенность «2» (Рис.54).

Лемма 23: Если на ***F***-модели $\mathbf{\Sigma`}$ определено отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by***, удовлетворяющее условию рефлексивности, то $\mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\mathbf{\Sigma`})\subseteq\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma`})$.

□479

Из условия транзитивности следует вложенность «3» (Рис.54).

Лемма 24: Пусть на ***F***-модели $\mathbf{\Sigma`}$ определено отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by***, удовлетворяющее условию транзитивности. Тогда $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma`})\subseteq\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$.

□481

Как следствие всех трёх вложений имеем основное утверждение.

Теорема 19: $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma) = \mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\Sigma`) = \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma`)$, где $\Sigma` = \cup_{\circ}\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)$ с отношением $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \boldsymbol{by}\ \Sigma`\ \boldsymbol{after}\ \sigma = \boldsymbol{safe\ in}\ \Sigma`\ \boldsymbol{after}\ \sigma$ для $\sigma \in \boldsymbol{SafeIn}(\Sigma`)$.

□481

Теорема 20: На классе пополненных ***F***-спецификаций отношение $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ является предпорядком.

□482

В то же время класс пополненных спецификаций эквивалентен классу всех спецификаций в том смысле, что описывает все возможные множества конформных реализаций в $\mathfrak{R}/\mathfrak{Q}$-семантике. Иными словами, мы можем считать не пополненные спецификации «лишними», поскольку на более узком классе пополненных спецификаций достигается та же спецификационная мощность.

Теорема 21: На классе спецификаций, являющихся объединениями конформных реализаций $\Sigma`$, где $\Sigma$ ***F***-спецификация с отношением $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \boldsymbol{by}\ \Sigma$, на которых задано отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \boldsymbol{by}\ \Sigma`\ \boldsymbol{after}\ \sigma = \boldsymbol{safe\ in}\ \Sigma`\ \boldsymbol{after}\ \sigma$ на безопасных трассах $\sigma \in \boldsymbol{SafeIn}(\Sigma`)$, отношение $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ эквивалентно вложенности $\mathfrak{R}$-трасс.

□482

Теперь рассмотрим случай LTS-моделей. Класс LTS-реализаций, конформных данной спецификации, не является множеством. Поэтому мы не можем применять Теорема 10: для объединения этих LTS-реализаций. Причина в том, что класс не может быть компонентом совокупности, а LTS – это совокупность. Если бы класс мог быть компонентом совокупности, мы могли бы определить LTS с классом состояний и объединение класса LTS. Но в объединении LTS из начального состояния ведут только τ-переходы. Если бы объединение конформных LTS-реализаций могло быть конформной LTS-реализацией, то в объединении конформных LTS-реализаций из начального

состояния вела бы бесконечная цепочка τ-переходов, то есть появилась бы трасса $\langle\Delta\rangle$. А тогда была бы не верна Теорема 10: для класса LTS без трассы $\langle\Delta\rangle$, и мы бы пришли к противоречию: объединение конформных LTS-реализаций не было бы конформной LTS-реализацией для спецификации без трасс $\langle\Delta\rangle$ и $\langle\gamma\rangle$. Эта ситуация является частным проявлением парадоксов, связанных с наивной теорией множеств.

Вместо этого мы можем взять объединение конформных трассовых моделей, которые образуют множество (как подмножество всех трассовых моделей) и построить по нему LTS с тем же множеством трасс. В этом случае пополнение LTS-спецификации $\mathbf{s}$ определяется как LTS $\boldsymbol{F2L}\circ\cup\circ\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}\circ\boldsymbol{F}(\mathbf{s})$ или $\boldsymbol{F2}_{\boldsymbol{compact}}\boldsymbol{L}\circ\cup\circ\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}\circ\boldsymbol{F}(\mathbf{s})$.

5.5. Проблема сохранения безопасности

Структура раздела:

1. Расширение класса безопасных реализаций
2. Сужение класса безопасных реализаций
3. Причины сужения класса безопасных реализаций

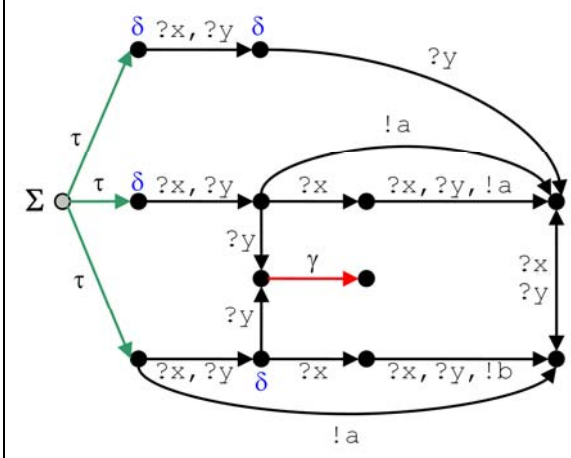
В отличие от конформных реализаций безопасные реализации, вообще говоря, не сохраняются при пополнении.

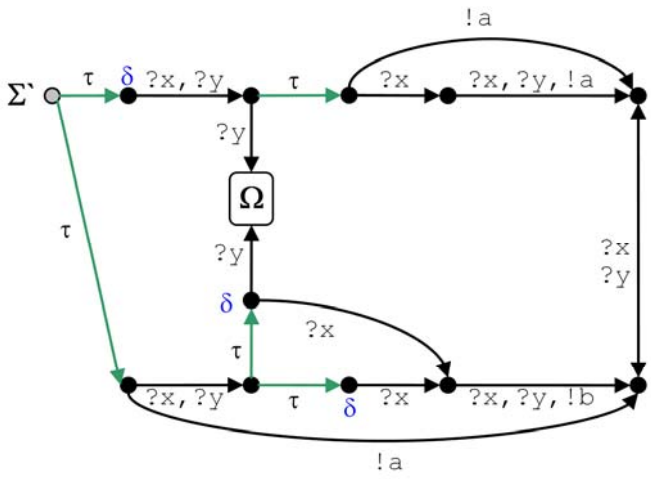
5.5.1. Расширение класса безопасных реализаций

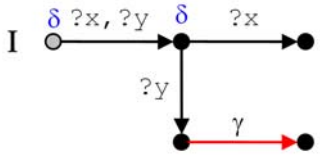
На Рис.55 показан пример, когда в результате пополнения спецификации $\boldsymbol{\Sigma}$ опасная реализация $\mathbf{I}$ становится безопасной. Здесь используется γδ-семантика: стимулы `?x` и `?y` передаются с помощью $\mathfrak{Q}$-кнопок “`{?x}`” и “`{?y}`”, а реакции `!a` и `!b` – с помощью одной кнопки “δ”=“`{!a,!b}`”. В

спецификации **Σ** трасса ⟨δ, ?x, δ⟩ до пополнения не продолжалась стимулом ?x, но продолжалась безопасным стимулом ?y. Поэтому реализация **I**, в которой после этой трассы стимул ?y разрушающий, оказывалась опасной.

Опасная реализация становится безопасной при пополнении







$\mathfrak{R}=\{\delta\}=\{\{!a,!b\}\}$, $\mathfrak{Q}=\{\{?x\},\{?y\}\}$, $\Sigma`=\cup\circ\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)$.

I ***~~safe for~~*** **Σ**, но **I** ***safe for*** **Σ`**

Рис.55.

После пополнения в спецификации **Σ`** трасса ⟨δ, ?x, δ⟩ исчезает: она не может быть в конформных реализациях, поскольку трасса ⟨δ, ?x, δ, ?x⟩ не может продолжаться ни какой-либо реакцией, ни стационарностью. Это объясняется тем, что её подтрассы не имеют общего продолжения: подтрасса ⟨δ, ?x, ?x⟩ продолжается только реакцией !a, а подтрасса ⟨?x, δ, ?x⟩ – только реакцией !b. В общем, это проявление общего правила о том, что пересечение моделей может быть неконвергентно (3.3.2): в данном случае пересекаются модели вида **Σ** ***after*** μ, где $\mu\in$ ***di***(⟨δ, ?x, δ⟩). Поэтому

реализация $\mathbf{I}$ становится безопасной для пополненной спецификации $\mathbf{\Sigma`}$: после подтрасс $\langle \delta, ?x \rangle$, $\langle ?x, \delta \rangle$ и $\langle ?x \rangle$ стимул `?y` разрушающий в спецификации $\mathbf{\Sigma`}$.

Расширение класса безопасных реализаций не является проблемой. Это можно считать даже полезным свойством: если до пополнения реализация была неконформна, поскольку опасна, то после пополнения мы можем эту неконформность проверять тестированием.

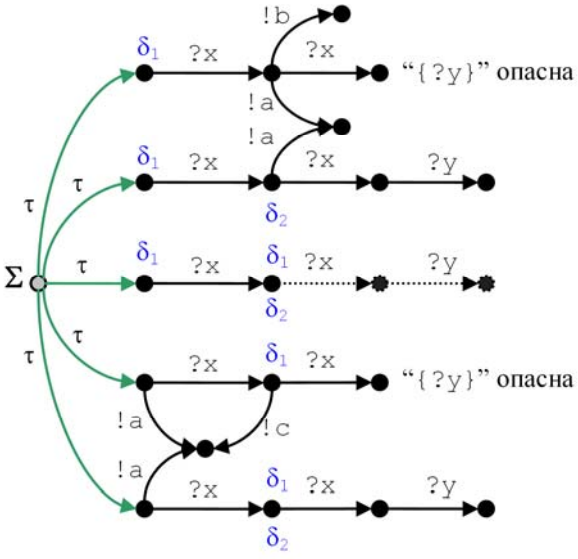
5.5.2. Сужение класса безопасных реализаций

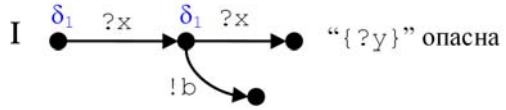
Сужение класса безопасных реализаций плохое свойство: реализации, которые мы раньше проверяли тестированием и обнаруживали их неконформность, теперь тестировать нельзя.

Один пример приведён на Рис.56. Здесь используется семантика для систем с мультиканалами стимулов и реакций (*MIOTS – Multi Input-Output Transition Systems*) [95,96] с двумя выходными каналами: 1) для реакции `!a` и 2) для двух реакций `!b` и `!c`. Соответственно, есть две частичные стационарности δ_1=`{!a}` и δ_2=`{!b,!c}`. Стимулы `?x` и `?y` передаются с помощью $\mathfrak{Q}$-кнопок “`{?x}`” и “`{?y}`”. В спецификации $\mathbf{\Sigma}$ трасса $\langle \delta_1, ?x, \delta_1 \rangle$ до пополнения не продолжалась стимулом `?x`, а каждая из подтрасс $\langle \delta_1, ?x \rangle$, $\langle ?x, \delta_1 \rangle$ и $\langle ?x \rangle$ продолжалась стимулом `?x`, после чего стимул `?y` был опасен, и кнопка “`{?y}`” не нажималась после этих подтрасс при безопасном тестировании. Поэтому реализация $\mathbf{I}$ с $\boldsymbol{F}$-трассами $\langle \delta_1, ?x, \delta_1, ?x, \{?y\} \rangle$ и $\langle \delta_1, ?x, \delta_1, !b \rangle$ была безопасной, хотя и неконформной: после трассы $\langle ?x, \delta_1 \rangle$ не разрешена реакция `!b`. В то же время реализация с трассой $\langle \delta_1, ?x, \delta_1, \delta_2, ?x, ?y \rangle$ была безопасной и конформной. Поэтому после

пополнения в спецификации **Σ`** появляется такая трасса, а реализация **I** становится опасной: теперь в спецификации **Σ`** есть трасса $\langle \delta_1, ?x, \delta_1, ?x \rangle$, после которой стимул `?y` безопасен.

Безопасная реализация становится опасной при пополнении (случай $\mathfrak{R}$-кнопок)





Пунктирные переходы появляются в объединении конформных реализаций **Σ`**.

`L={?x,?y,!a,!b,!c}`,
$\mathfrak{Q}$`={{?x},{?y}}`,
$\mathfrak{R}$`={`δ_1`,`δ_2`}`,
где δ_1`={!a}` и δ_2`={!b,!c}`.

I ***safe for*** **Σ**, но **I** ~~***safe for***~~ **Σ`**

Рис.56.

Другой пример приведён на Рис.57. Здесь модель **Σ** совпадает с объединением конформных реализаций **Σ`**. Однако в спецификации **Σ** $\mathfrak{Q}$-кнопка "`{a,b}`" опасна после любой трассы, а в **Σ`** безопасна, поскольку в объединении конформных реализаций мы задаём отношение ***safe by*** **Σ`** ***after*** σ = ***safe in*** **Σ`** ***after*** σ на безопасных трассах σ∈***SafeIn***(**Σ`**). Поскольку в реализации **I** после любой трассы есть $\mathfrak{Q}$-отказ `{a,b}`, она становится опасной.

Безопасная реализация становится опасной при пополнении (случай $\mathfrak{Q}$-кнопок)	
$\Sigma`=\Sigma$  I 	L={a,b,c}, $\mathfrak{R}$={{a}}, $\mathfrak{Q}$={{a,b},{b,c}} $\forall\sigma$ {a,b} ~~***safe by***~~ **Σ** ***after*** σ & {b,c} ***safe by*** **Σ** ***after*** σ $\forall\sigma$ {a,b} ***safe by*** **Σ`** ***after*** σ & {b,c} ***safe by*** **Σ`** ***after*** σ **I** ***safe for*** **Σ** **I** ~~***safe for***~~ **Σ`**
Рис.57.	

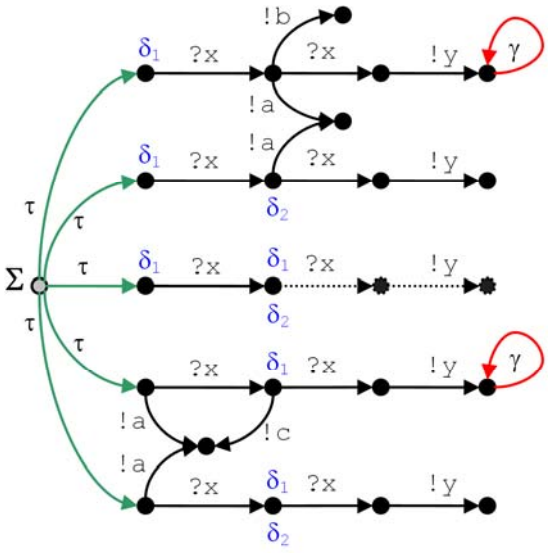
5.5.3. Причины сужения класса безопасных реализаций

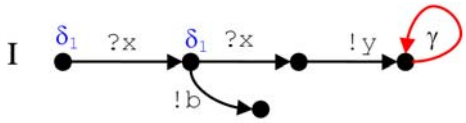
Рассмотрим более подробно причины, по которым безопасная реализация после пополнения становится опасной.

В примере на Рис.56 для сохранения безопасности реализации **I** нам следовало бы объявить стимул ?y опасным после трассы $\langle\delta_1,$?x,$\delta_1,$?x$\rangle$, оставив его безопасным после трассы $\langle\delta_1,$?x,$\delta_1,\delta_2,$?x$\rangle$. Однако в этом случае будет нарушено второе правило протокола: первая из этих трасс продолжается в пополненной спецификации стимулом ?y, который разрешается неразрушающей кнопкой "{?y}", но не разрешается безопасными кнопками.

На Рис.58 изображён пример, отличающийся от примера на Рис.56 тем, что при пополнении после трассы $\langle\delta_1,$?x,$\delta_1,$?x$\rangle$ становится безопасной не $\mathfrak{Q}$-кнопка "{?y}", а $\mathfrak{R}$-кнопка частичной стационарности δ_3={!y}. Здесь для сохранения безопасности реализации **I** нам следовало бы объявить реакцию !y опасной после трассы $\langle\delta_1,$?x,$\delta_1,$?x$\rangle$, оставив её безопасной после трассы $\langle\delta_1,$?x,$\delta_1,\delta_2,$?x$\rangle$. Однако в этом случае будет нарушено не только второе правило протокола, но ещё и первое правило: после первой из этих трасс $\mathfrak{R}$-кнопка частичной стационарности "δ_3" будет не разрушающей, но опасной.

Безопасная реализация становится опасной при пополнении (случай $\mathfrak{R}$-кнопок)





Пунктирные переходы появляются в объединении конформных реализаций $\mathbf{\Sigma`}$.

L={?x,!y,!a,!b,!c},

$\mathfrak{Q}$={{?x}},

$\mathfrak{R}$={δ_1,δ_2,δ_3},

где δ_1={!a}, δ_2={!b,!c} и δ_3={!y}.

I ***safe for*** **Σ**

I ~~***safe for***~~ **Σ`**

Рис.58.

Первое правило протокола говорит об эквивалентности неразрушаемости и безопасности для $\mathfrak{R}$-кнопок. Для сохранения безопасности при пополнении нам пришлось бы пожертвовать импликацией: неразрушаемость⇒безопасность.

Второе правило протокола имеет целью использовать спецификацию по максимуму, поэтому его нарушение приводит к тому, что в спецификации появляется слишком много «лишних» трасс, то есть не используемых для определения и тестирования конформности. В то же время нарушение этого правила в рассматриваемых примерах частичное: хотя действие z, продолжающее трассу μ и разрешаемое неразрушающей после μ кнопкой, не разрешается безопасными после μ кнопками, оно разрешается безопасной кнопкой после некоторой трассы $\mu` \in \boldsymbol{di}(\mu)$, получающейся из $\mu$ удалением отказов и вставкой «обязательных» отказов. В наших примерах $\mu=\langle\delta_1,?x,\delta_1,?x\rangle$, z=?y или z=!y, и $\mu`=\langle\delta_1,?x,\delta_1,\delta_2,?x\rangle$.

Таким образом, для сохранения безопасности мы могли бы ослабить правила протокола. Можно даже разрешить объявлять безопасной после трассы μ $\mathfrak{Q}$-кнопку, когда μ не продолжается действиями, разрешаемыми этой кнопкой. Мы уже отмечали, что в этом случае любая реализация, имеющая

трассу μ, оказывается неконформной. Но само по себе это не страшно, так как и без этого спецификация могла иметь неконформные трассы. После этого третье правило для $\mathfrak{Q}$-кнопок уже не отличается от ослабленного первого правила для $\mathfrak{R}$-кнопок.

В целом можно определить следующие ослабленные правила протокола:

$\forall \sigma \in \mathbf{\Sigma}^{\downarrow} L_{\mathfrak{R}}\ \ \forall P \in \mathfrak{R} \cup \mathfrak{Q}\ \ \forall z \in L$

1) P ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma \Rightarrow \forall u \in P\ \ \sigma \cdot \langle u, \gamma \rangle \notin \mathbf{\Sigma}\ \ \&\ \ \sigma \cdot \langle \Delta \rangle \notin \mathbf{\Sigma}$,

2) $\sigma \cdot \langle z \rangle \in \mathbf{\Sigma}\ \ \&\ \ \exists T \in \mathfrak{R} \cup \mathfrak{Q}\ \ z \in T\ \ \&\ \ \forall u \in T\ \ \sigma \cdot \langle u, \gamma \rangle \notin \mathbf{\Sigma}\ \ \&\ \ \sigma \cdot \langle \Delta \rangle \notin \mathbf{\Sigma}$

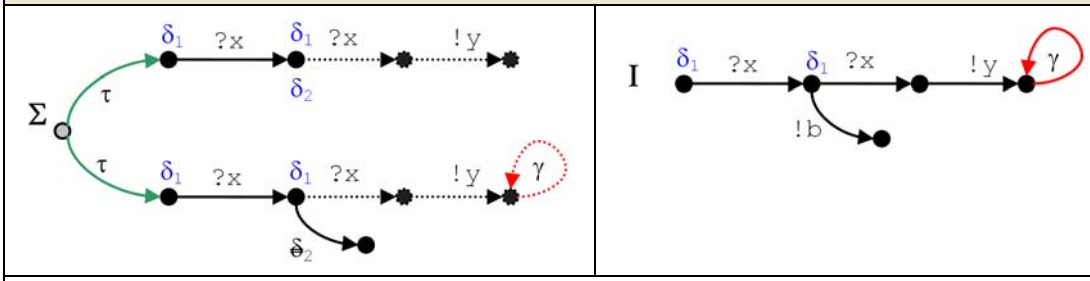
$\Rightarrow \exists \mu \in \boldsymbol{di}(\sigma)\ \ \exists R \in \mathfrak{R} \cup \mathfrak{Q}\ \ z \in R\ \ \&\ \ R$ ***safe by*** $\mathbf{\Sigma}$ ***after*** μ.

Можно предположить, что с такими правилами удалось бы не только сохранить при пополнении класс конформных, но и не сузить класс безопасных реализаций (по крайней мере, для случаев аналогичных примерам на Рис.56 и Рис.58). Однако сами эти правила вызывают обоснованные сомнения. Хотя мы и говорим, что спецификация – это не только модель (трассовая или LTS), но и определённое на ней отношение ***safe by***, всё же имеется в виду, что такое отношение можно задавать единообразным способом на всех моделях как некое «допущение безопасности» по аналогии с «допущением полноты». Интуитивно хотелось бы, чтобы безопасность кнопки после трассы зависела только от того, какими действиями эта трасса продолжается в спецификации и какие из них разрушающие. В наших примерах трассы $\langle \delta_1, ?x, \delta_1, ?x \rangle$ и $\langle \delta_1, ?x, \delta_1, \delta_2, ?x \rangle$ имеют одинаковые продолжения, и поэтому безопасность кнопок после них должна была бы быть одинаковой. Кроме того, нарушение первого правила особенно трудно объяснить: если неразрушающая $\mathfrak{R}$-кнопка объявляется опасной, то в чём же эта опасность заключается?

Другое решение проблемы сохранения безопасности заключается в том, чтобы «разделить» трассы $\langle \delta_1, ?x, \delta_1, ?x \rangle$ и $\langle \delta_1, ?x, \delta_1, \delta_2, ?x \rangle$. В LTS

трасса $\langle \delta_1, ?x, \delta_1 \rangle$ заканчивалась бы не в одном, а в двух состояниях: в одном состоянии s_1 был бы отказ δ_2, а в другом состоянии s_2 не было бы отказа δ_2. По конвергентности модели, в состоянии s_2 должна выдаваться реакция из δ_2. Но дело-то как раз в том, что ни одна из реакций, оставляющих трассу согласованно, то есть реакций $!b$ и $!c$ не может выдаваться в конформных реализациях после трассы $\langle \delta_1, ?x, \delta_1 \rangle$ (пересечение моделей неконвергентно). Поэтому мы не можем определять в спецификации переход по этим реакциям из состояния s_2. Выход в том, чтобы добавить в алфавит спецификации и в каждую $\mathfrak{R}$-кнопку "R" фиктивное внешнее действие «не-отказ» $\not{R}$, понимаемое как «не R», т.е. «отсутствие отказа R». Тогда в состоянии s_2 проводится переход по действию $\not{R}$ (Рис.59).

Пополнение с действиями «отсутствие $\mathfrak{R}$-отказа»



Пунктирные переходы появляются в пополнении $\mathbf{\Sigma}`$.
$L=\{?x, !y, !a, !b, !c, \not\delta_1, \not\delta_2, \not\delta_3\}$, $\mathfrak{Q}=\{\{?x\}\}$, $\mathfrak{R}=\{\delta_1^{\rightarrow}, \delta_2^{\rightarrow}, \delta_3^{\rightarrow}\}$, где $\delta_1^{\rightarrow}=\{!a, \not\delta_1\}$, $\delta_2^{\rightarrow}=\{!b, !c, \not\delta_2\}$ и $\delta_3^{\rightarrow}=\{!y, \not\delta_3\}$.
I ***safe for*** $\mathbf{\Sigma}$ и **I** ***safe for*** $\mathbf{\Sigma}`$.

Рис.59.

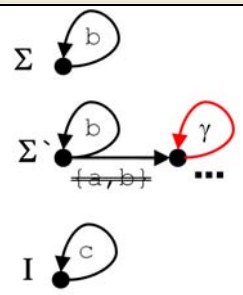
Теперь реализация **I**, которая была безопасна до пополнения, остаётся безопасной после пополнения. Конечно, такая безопасная реализация заведомо неконформна: после трассы $\langle \delta_1, ?x, \delta_1 \rangle$ в ней есть реакция $!b$, а требуется не-отказ $\not\delta_2$. Какое же преимущество мы получили при таком пополнении с $\not\delta_2$? Теперь мы можем тестировать реализацию **I** и обнаруживать её

неконформность, как это было до пополнения и как нельзя делать после пополнения без δ_2. Неконформность проявляется, когда после трассы $\langle \delta_1, ?x, \delta_1 \rangle$ нажимается кнопка "δ_2": это первый тест. Но есть и второй тест, когда после этой трассы нажимается кнопка "?x". Для пополнения без δ_2 после получения стимула ?x во втором тесте можно было нажимать кнопку "!y" (или "?y" для примера на Рис.56). Это налагало на реализацию лишние ограничения по безопасности, и реализация $\mathbf{I}$ оказывалась опасной. Первый тест обнаруживал неконформность, а второй вызывал разрушение (или ненаблюдаемый отказ в примере на Рис.56). Для исходной спецификации и пополнения с δ_2 кнопка "!y" (или "?y") опасна и не нажимается во втором тесте, и потому реализация $\mathbf{I}$ безопасна, хотя и неконформна. Теперь эти два теста можно прогонять в любом порядке.

В общем случае для такого «разделения» трасс мы продолжим в исходной спецификацию каждую трассу μ каждым не-отказом $\langle ₽ \rangle$, если трасса $\mu \cdot \langle ₽ \rangle$ остаётся согласованной и допустимой. Ни одна «старая» реализация не может иметь трассу с ₽.

В примере на Рис.57 для сохранения безопасности реализации $\mathbf{I}$ нам было бы достаточно сохранить в объединении конформных реализаций $\mathbf{\Sigma`}$ то отношение ***safe by***, которое было в исходной спецификации $\mathbf{\Sigma}$ (в данном примере она совпадает с $\mathbf{\Sigma`}$). Однако такое отношение не будет совпадать с отношением ***safe in*** на безопасных трассах. Причина в том, что для отношения ***safe by*** после трассы $\mathbf{\Omega}$-кнопка "Q" может быть опасной, хотя и не разрушающей, а трасса не продолжается отказом Q. Поэтому другое решение заключается в том, чтобы сделать такую $\mathbf{\Omega}$-кнопку разрушающей, то есть продолжить трассу некоторым разрушающим действием $z \in Q$. Проблема в том, что все действия из Q могут оказаться безопасными (как в данном примере для Q={a,b}).

Выход опять в том, чтобы добавить в алфавит спецификации и в каждую $\mathfrak{Q}$-кнопку “Q” фиктивное внешнее действие «не-отказ» $\ominus$, понимаемое как «не Q», т.е. «отсутствие отказа Q» (Рис.60).

Пополнение с действиями «отсутствие $\mathfrak{Q}$-отказа»	
	L={a,b,c}, $\mathfrak{R}$={{a}}, $\mathfrak{Q}$={{a,b},{b,c}} ∀σ {a,b} ~~***safe by***~~ **Σ** ***after*** σ & {b,c} ***safe by*** **Σ** ***after*** σ ∀σ {a,b} ~~***safe by***~~ **Σ`** ***after*** σ & {b,c} ***safe by*** **Σ`** ***after*** σ **I** ***safe for*** **Σ** **I** ***safe for*** **Σ`**
Рис.60.	

5.6. Пополнение с не-отказами

Структура раздела:

1. Основные идеи пополнения с не-отказами
2. Формальное определение пополнения с не-отказами
3. Теорема о пополнении с не-отказами
4. Алгоритмизация
5. Частные случаи δ-, γδ- и βγδ-семантик

В этом разделе мы дадим конструктивное (индуктивное) определение пополнения с не-отказами и докажем сохранение класса конформных и не сужение класса безопасных реализаций. Такое преобразование пополнения, в отличие от объединения конформных реализаций, при определённых ограничениях поддаётся алгоритмизации, то есть может быть переформулировано как алгоритмическое преобразование.

5.6.1. Основные идеи пополнения с не-отказами

Для исходной спецификации $\mathbf{\Sigma}$ пополнение с не-отказами $\boldsymbol{Comp}(\mathbf{\Sigma})$ совмещает введение не-отказов и взятие трасс конформных реализаций, причём не всех, а только необходимых для безопасности и конформности. Оно основано на следующих идеях:

1. В пополнении $\boldsymbol{Comp}(\mathbf{\Sigma})$ не должно быть $\mathfrak{Q}$-отказов (по крайней мере, после безопасных трасс). Это необходимо для того, чтобы классы конформных реализаций совпадали в $\mathfrak{R}/\mathfrak{Q}$- и $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантиках. В частности, это необходимо для рефлексивности отношения ***saco***.
2. Отношение ***safe by*** отождествляется с отношением ***safe in***. Это также необходимо для того, чтобы классы конформных реализаций совпадали в $\mathfrak{R}/\mathfrak{Q}$- и $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантиках. Такое отношение ***safe by*** удовлетворяет всем трём правилам протокола, если в спецификации нет $\mathfrak{Q}$-отказов. Тем самым, любая кнопка опасна тогда и только тогда, когда она разрушающая.
3. Пополнение $\boldsymbol{Comp}(\mathbf{\Sigma})$ основано на трассах, которые можно назвать *финальными*. Это безопасные трассы и безопасные трассы, продолженные разрушающим действием (действием и далее разрушением) или дивергенцией. Такие продолжения позволяют указывать, какие кнопки безопасны, а какие опасны после безопасных трасс. Все остальные трассы, которые могут быть в конформных реализациях, можно удалить с сохранением безопасных и конформных реализаций. Это требование носит чисто оптимизационный характер: мы просто не строим «лишние» трассы.
4. Все безопасные трассы исходной спецификации $\mathbf{\Sigma}$ сохраняются в пополнении $\boldsymbol{Comp}(\mathbf{\Sigma})$. Это необходимо для того, чтобы не обнаруживать неконформность уже построенной трассы «задним числом». В примере на Рис.15 спецификация содержит безопасную трассу $\langle \delta, ?x, \delta \rangle$, которой не может быть в конформных реализациях. Причина в том, что пересечение

деревьев после подтрасс $\langle \delta, ?x, ?x \rangle$ и $\langle ?x, \delta, ?x \rangle$ неконвергентно. Мы могли бы обнаруживать эту неконвергентность в процессе построения трасс пополнения, и отбрасывать неконвергентные трассы. Проблема в том, что заранее неизвестно, когда будет обнаружена неконформность трассы, то есть как долго нужно строить её продолжение, чтобы обнаружить неконвергентность. В примере на Рис.15 мы могли бы заменить трассы $\langle \delta, ?x, ?x \rangle$ и $\langle ?x, \delta, ?x \rangle$ на трассы $\langle \delta, ?x, ?x, \ldots, ?x \rangle$ и $\langle ?x, \delta, ?x, \ldots, ?x \rangle$, где многоточие означает последовательность действий любой наперёд заданной длины. Тем самым, для некоторых уже построенных трасс возможность такого «отката назад» не позволяет генерировать тесты ни в какой момент времени, поскольку неизвестно, окажется ли трасса конформной или нет.

5. Безопасная трасса μ пополнения $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ может не принадлежать исходной спецификации $\boldsymbol{\Sigma}$ (в рассмотренном выше примере это трасса $\langle \delta, ?x, \delta, ?x \rangle$) , но должна иметь ***drt***-подтрассу, безопасную в $\boldsymbol{\Sigma}$. В противном случае мы усилили бы требования конформности. Трасса μ безопасно продолжается действием или $\mathfrak{R}$-отказом u тогда и только тогда, когда выполняются следующие три требования:

1. Продолженная трасса $\mu \cdot \langle u \rangle$ согласована (несогласованность возникает, если u – это действие, принадлежащее некоторому отказу в постфиксе трассы μ). Это требование необходимо для согласованности пополнения.
2. Существует трасса $\lambda \in \boldsymbol{drt}(\mu)$, безопасная в $\boldsymbol{\Sigma}$, после которой символ u безопасен в $\boldsymbol{\Sigma}$. Иными словами, существует такая трасса $\lambda \in \boldsymbol{drt}(\mu)$, что продолженная трасса $\lambda \cdot \langle u \rangle$ тестовая. Это необходимо для того, чтобы не усиливать конформность.

3. Каждая трасса $\lambda \in \boldsymbol{drt}(\mu)$, безопасная в $\boldsymbol{\Sigma}$, после которой символ u безопасен в $\boldsymbol{\Sigma}$, продолжается символом u в $\boldsymbol{\Sigma}$. Иными словами, не существует такой трассы $\lambda \in \boldsymbol{drt}(\mu)$, что продолженная трасса $\lambda \cdot \langle u \rangle$ тестовая, но не принадлежит $\boldsymbol{\Sigma}$. Это необходимо для того, чтобы не ослаблять конформность.

6. Для того, чтобы можно было выполнить требования предыдущих пунктов, необходимо добавить продолжения безопасных трасс не-отказами. Мы будем продолжать трассу μ не-отказом $\Rightarrow$ всегда, когда это не приводит к несогласованности, то есть когда в постфиксе отказов трассы μ нет отказа P. Очевидно, это не влияет на безопасность или конформность реализаций без не-отказов. За не-отказом всегда будет следовать a) разрушение или b) дивергенция. Случай a). Не-отказ $\Rightarrow$ разрушающий, если после каждой трассы $\lambda \in \boldsymbol{drt}(\mu)$, безопасной в $\boldsymbol{\Sigma}$, отказ P опасный. Иными словами, не существует такой трассы $\lambda \in \boldsymbol{drt}(\mu)$, что продолженная трасса $\lambda \cdot \langle u \rangle$ тестовая. В этом случае наличие или отсутствие остальных возможных в конформных реализациях продолжений трассы μ действиями из P или самим отказом P (если это $\mathfrak{R}$-отказ) не влияет на безопасность и конформность. Из соображений оптимизации мы считаем такие продолжения «лишними» и не включаем в пополнение. Случай b). Это случай, когда после некоторой трассы $\lambda \in \boldsymbol{drt}(\mu)$, безопасной в $\boldsymbol{\Sigma}$, отказ P безопасный. Иными словами, существует такая трасса $\lambda \in \boldsymbol{drt}(\mu)$, что продолженная трасса $\lambda \cdot \langle u \rangle$ тестовая. Тогда не-отказ $\Rightarrow$ безопасный, и мы продолжаем его дивергенцией. Любые продолжения после любого не-отказа, которые могут быть в конформных реализациях, нас не интересуют, поскольку мы ставим задачу сохранения только таких безопасных и конформных реализаций, в которых нет не-отказов.

Можно заметить, что мы говорим о ***drt***-трассах, а не ***di***-трассах, поскольку в пополнении ***i***-операция не вставляет никаких отказов, которые нельзя было бы вставить с помощью ***rt***-операций. Это объясняется тем, что любую безопасную трассу мы продолжаем каждым не-отказом $₽$, отсутствующим в постфиксе отказов трассы. Следовательно, ***i***-операция должна вставлять только такие отказы, которые есть в постфиксе трассы, а это можно всегда сделать с помощью ***rt***-операций.

5.6.2. Формальное определение пополнения с не-отказами

Определение 70: Будем считать, что каждому множеству внешних действий $P \subseteq Z$ поставлен во взаимно-однозначное соответствие праэлемент $₽ \notin Z$, который будем называть *не-отказом*. Не-отказ будем считать дополнительным внешним действием, отсутствующим в исходных спецификациях и реализациях. Объединение $Z^{\#}=Z \cup \{₽ \mid P \subseteq Z\}$ будем считать новым универсумом внешних действий.

□

Заметим, что все определения, которые были сформулированы, и все утверждения, которые были доказаны, для алфавита $L \subseteq Z$ остаются верными для алфавита $L` \subseteq Z^{\#}$, поскольку нигде не использовались какие-либо свойства универсума внешних действий (кроме того, что все внешние действия праэлементы).

Нам нужно формально для каждого $\mathfrak{R} \cup \mathfrak{Q}$-отказа P добавить не-отказ $₽$ в отказ P, то есть заменить P на $P \cup \{₽\}$. Мы будем обозначать это преобразование стрелкой в верхнем индексе “$\rightarrow$” и использовать постфиксную запись. Оно естественно распространяется на семейства отказов (в частности, $\mathfrak{R}$ и $\mathfrak{Q}$), трассы (действия и отказы не из $\mathfrak{R} \cup \mathfrak{Q}$ не меняются), множества трасс (в частности, модели) и семейства множеств трасс (в частности, классы

моделей). Также мы определим обратное преобразование "$^{\leftarrow}$". Объединение алфавита L со всеми не-отказами $\Rightarrow$, где $P \in \mathfrak{R} \cup \mathfrak{Q}$, будем обозначать $L^{\Rightarrow}$.

Преобразование "$^{\rightarrow}$", применённое к ***F***-модели $\mathbf{I}$, вообще говоря, не даёт ***F***-модель, поскольку, например, исчезает пустой отказ. Для получения ***F***-модели можно было бы взять $\mathfrak{R} \cup \mathfrak{Q}$-проекцию модели $\mathbf{I}$, применить к ней преобразование "$^{\rightarrow}$" (легко показать, что в результате получится $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$-модель), а потом выполнить расширение до ***F***-модели в алфавите $L^{\Rightarrow}$. Соответствующее преобразование будем обозначать двойной стрелкой в верхнем индексе "$^{\Rightarrow}$" и использовать постфиксную запись: $\mathbf{I}^{\Rightarrow} = \boldsymbol{Ext}((\mathbf{I}^{\downarrow L_{\mathfrak{R} \cup \mathfrak{Q} \Delta \gamma}})^{\rightarrow})$. Для LTS преобразование "$^{\Rightarrow}$" иожно определить как простое изменение алфавита. Чтобы преобразовать трассовую ***F***-модель $\mathbf{I}$, нужно взять соответствующую ей LTS $\boldsymbol{F2L}(\mathbf{I})$, преобразовать её в $\boldsymbol{F2L}(\mathbf{I})^{\Rightarrow}$ и после этого взять множество ***F***-трасс полученной LTS $\boldsymbol{F}(\boldsymbol{F2L}(\mathbf{I})^{\Rightarrow})$.

С помощью этих преобразований мы перейдём от $\mathfrak{R}/\mathfrak{Q}$-семантики к $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$-семантике и от модели $\mathbf{I}$ к модели $\mathbf{I}^{\Rightarrow}$.

Определение 71:

- Для алфавита определим: $L^{\Rightarrow} =_{def} L \cup \{\Rightarrow \mid P \in \mathfrak{R} \cup \mathfrak{Q}\}$.
- Определим преобразование "$^{\rightarrow}$" добавления не-отказов:
  - $P \in \mathfrak{R} \cup \mathfrak{Q}$: $P^{\rightarrow} =_{def} P \cup \{\Rightarrow\}$,
  - $P \in \mathcal{P}(L) \setminus (\mathfrak{R} \cup \mathfrak{Q})$: $P^{\rightarrow} =_{def} P$,
  - $z \in L$: $z^{\rightarrow} =_{def} z$.
- Определим обратное преобразование "$^{\leftarrow}$" удаления не-отказов:
  - $R \subseteq Z^{\#}$: $R^{\leftarrow} =_{def} R \cap Z$,
  - $z \in Z^{\#}$: $z^{\leftarrow} =_{def} z$.

□

Преобразования “$\rightarrow$” и “$\leftarrow$” естественно распространяются:

на семейства отказов как поэлементные преобразования семейств:

- $\mathfrak{R}^{\rightarrow} =_{def} \{P^{\rightarrow} \mid P \in \mathfrak{R}\}$, $\mathfrak{Q}^{\rightarrow} =_{def} \{P^{\rightarrow} \mid P \in \mathfrak{Q}\}$,
- $\mathfrak{R}^{\leftarrow} =_{def} \{P^{\leftarrow} \mid P \in \mathfrak{R}\}$, $\mathfrak{Q}^{\leftarrow} =_{def} \{P^{\leftarrow} \mid P \in \mathfrak{Q}\}$;

на последовательности, то есть трассы, как композиция отображений:

- $\mu^{\rightarrow} =_{def} \langle \mu(i)^{\rightarrow} \mid i \in [1..|\mu|] \rangle$,
- $\mu^{\leftarrow} =_{def} \langle \mu(i)^{\leftarrow} \mid i \in [1..|\mu|] \rangle$;

на множества трасс, в частности на модели, как поэлементные преобразования множеств:

- $\mathbf{\Sigma}^{\rightarrow} =_{def} \{\mu^{\rightarrow} \mid \mu \in \mathbf{\Sigma}\}$,
- $\mathbf{\Sigma}^{\leftarrow} =_{def} \{\mu^{\leftarrow} \mid \mu \in \mathbf{\Sigma}\}$;

на семейства множеств трасс как поэлементные преобразования семейств:

- $\mathfrak{S}^{\rightarrow} =_{def} \{\mathbf{\Sigma}^{\rightarrow} \mid \mathbf{\Sigma} \in \mathfrak{S}\}$,
- $\mathfrak{S}^{\leftarrow} =_{def} \{\mathbf{\Sigma}^{\leftarrow} \mid \mathbf{\Sigma} \in \mathfrak{S}\}$.

Хотя домен преобразования “$\leftarrow$” шире области значений преобразования “$\rightarrow$”, на этой области значений оно является обратным преобразованием:

$\mathfrak{R}^{\rightarrow\leftarrow}=\mathfrak{R}$, $\mathfrak{Q}^{\rightarrow\leftarrow}=\mathfrak{Q}$, $\mu^{\rightarrow\leftarrow}=\mu$, $\mathbf{\Sigma}^{\rightarrow\leftarrow}=\mathbf{\Sigma}$, $\mathfrak{S}^{\rightarrow\leftarrow}=\mathfrak{S}$.

Определение 72:

- Определим преобразование “$\Rightarrow$” для LTS: для $\mathbf{S}=LTS(V_{\mathbf{S}}, L, E_{\mathbf{S}}, s_0)$ определим $\mathbf{S}^{\Rightarrow} =_{def} LTS(V_{\mathbf{S}}, L^{\Rightarrow}, E_{\mathbf{S}}, s_0)$.
- Определим преобразование “$\Rightarrow$” для ***F***-моделей: для $\mathbf{I} \in \boldsymbol{FMODEL}(L)$ определим $\mathbf{I}^{\Rightarrow}=\boldsymbol{F}(\boldsymbol{F2L}(\mathbf{I})^{\Rightarrow})$.

□

Лемма 25: Пусть задана ***F***-модель $\mathbf{I}$. Преобразование "$\Rightarrow$" даёт ***F***-модель $\mathbf{I}^{\Rightarrow}$ в алфавите $\mathtt{L}^{\Rightarrow}$, которая преобразованием "$\leftarrow$" переводится в исходную модель: $\mathbf{I}^{\Rightarrow\leftarrow}=\mathbf{I}$.

□482

Теперь введём ряд вспомогательных обозначений.

Определение 73: Пусть задана ***F***-спецификация $\mathbf{\Sigma}$.

- Обозначим множество тестовых трасс, пополненное продолжениями безопасных трасс безопасными не-отказами:

  $\boldsymbol{tt}^{\#}(\mathbf{\Sigma}) =_{\text{def}} \boldsymbol{tt}(\mathbf{\Sigma})$

  $\cup\{\lambda\cdot\langle ₽\rangle \mid \lambda\in \boldsymbol{SafeBy}(\mathbf{\Sigma}) \;\&\; \mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q} \;\&\; \mathtt{P}\ \boldsymbol{safe\ by}\ \mathbf{\Sigma}\ \boldsymbol{after}\ \lambda\}$.

- Для каждой $\mathfrak{R}$-трассы $\mu\in \mathtt{L}_{\mathfrak{R}}^{\ *}$ обозначим:

  - множество действий, $\mathfrak{R}$-отказов и не-отказов, безопасных после трассы $\mu^{\rightarrow}$:

    $\boldsymbol{safe}(\mu) =_{\text{def}} \{\mathtt{u}\in \mathtt{L}_{\mathfrak{R}}\cup\{₽ \mid \mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}\} \mid \exists\lambda\in \boldsymbol{drt}(\mu)\ \ \lambda\cdot\langle \mathtt{u}\rangle\in \boldsymbol{tt}^{\#}(\mathbf{\Sigma})\}$.

  - множество не-отказов, *конформных и опасных* после трассы $\mu^{\rightarrow}$:

    $\boldsymbol{G}(\mu) =_{\text{def}} \{₽ \mid \mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q} \;\&\; \mu^{\rightarrow}\cdot\langle ₽\rangle$ согласована $\;\&\; ₽\notin \boldsymbol{safe}(\mu)\}$,

  - множество не-отказов, *конформных и безопасных* после трассы $\mu^{\rightarrow}$:

    $\boldsymbol{D}(\mu) =_{\text{def}} \{₽ \mid \mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q} \;\&\; \mu^{\rightarrow}\cdot\langle ₽\rangle$ согласована $\;\&\; ₽\in \boldsymbol{safe}(\mu)\}$,

  - условие конформности трассы $\mu^{\rightarrow}$:

    $\mu\ \boldsymbol{conf} =_{\text{def}} \neg(\exists\lambda\in \boldsymbol{drt}(\mu)\ \ \exists\kappa\ \ \exists\mathtt{u}\ \ \kappa\cdot\langle \mathtt{u}\rangle\le\lambda \;\&\; \kappa\cdot\langle \mathtt{u}\rangle\in \boldsymbol{tt}^{\#}(\mathbf{\Sigma})\setminus\mathbf{\Sigma})$,

  - множество действий и $\mathfrak{R}$-отказов, *конформных и безопасных* после трассы $\mu^{\rightarrow}$:

    $\boldsymbol{U}(\mu) =_{\text{def}} \{\mathtt{u}\in \mathtt{L}_{\mathfrak{R}} \mid \mu\cdot\langle \mathtt{u}\rangle$ согласована $\;\&\; \mathtt{u}\in \boldsymbol{safe}(\mu) \;\&\; \mu\cdot\langle \mathtt{u}\rangle\ \boldsymbol{conf}\}$.

□

Здесь можно сделать ряд замечаний о возможной оптимизации вычислений.

Во-первых, в определении $\boldsymbol{U}(\mu)$ (конформные и безопасные продолжения действиями и $\mathfrak{R}$-отказами) требуется согласованность трассы $\mu\cdot\langle u\rangle$, а не трассы $\mu^{\rightarrow}\cdot\langle u^{\rightarrow}\rangle$. Но для $\mu\in \mathtt{L}_{\mathfrak{R}}{}^{*}$ и $u\in \mathtt{L}_{\mathfrak{R}}$ эти согласованности эквивалентны.

Во-вторых, если трасса $\mu^{\rightarrow}$ согласована, то для проверки согласованности трассы $\mu^{\rightarrow}\cdot\langle u^{\rightarrow}\rangle$, где a) $u=z\in \mathtt{L}$ действие, или b) $u=P\in\mathfrak{R}$ отказ, или c) $u=\not\in$ не-отказ, достаточно проверить согласованность продолжения символом u: a) $z\notin\cup\circ\boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ или c) $P\notin\boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ (в случае b) трасса $\mu^{\rightarrow}\cdot\langle P^{\rightarrow}\rangle$ всегда согласована).

В-третьих, если для трассы $\mu^{\rightarrow}$ выполнено условие конформности, то проверка этого условия для трассы $\mu^{\rightarrow}\cdot\langle u^{\rightarrow}\rangle$, то есть условия $\mu\cdot\langle u\rangle$ ***conf***, не требует рассмотрения всех трасс $\lambda\in\boldsymbol{drt}(\mu\cdot\langle u\rangle)$, а только таких, в которых остаётся символ u.

Теперь определим преобразование, которое по исходной спецификации $\boldsymbol{\Sigma}$ строит множество финальных трасс. Далее мы покажем, что ***d***-замыкание этого множества является $\mathfrak{R}$-проекцией искомого пополнения спецификации.

Определение 74: Пусть задана ***F***-спецификация $\boldsymbol{\Sigma}$. Определим преобразование $\boldsymbol{final}(\boldsymbol{\Sigma})=\boldsymbol{\Sigma_0}\cup\boldsymbol{\Sigma_1}$, где множества $\boldsymbol{\Sigma_0}$ и $\boldsymbol{\Sigma_1}$ – наименьшие множества трасс в алфавите $\mathtt{L}^{\Rightarrow}$, порождаемые следующими правилами вывода: $\forall\mu\in \mathtt{L}_{\mathfrak{R}}{}^{*}$ $\forall P\in\mathfrak{R}\cup\mathfrak{Q}$ $\forall u\in \mathtt{L}_{\mathfrak{R}}$

1. $\langle\gamma\rangle\in\boldsymbol{\Sigma}$ $\quad\vdash\quad \epsilon\in\boldsymbol{\Sigma_1}$ & $\langle\gamma\rangle\in\boldsymbol{\Sigma_1}$,

2. $\langle\gamma\rangle\notin\boldsymbol{\Sigma}$ $\quad\vdash\quad \epsilon\in\boldsymbol{\Sigma_0}$,

3. $\mu^{\rightarrow}\in\boldsymbol{\Sigma_0}\ \&\ ₽\in\boldsymbol{G}(\mu)\ \vdash\ \mu^{\rightarrow}\cdot\langle ₽\rangle\in\boldsymbol{\Sigma_1}\ \&\ \mu^{\rightarrow}\cdot\langle ₽,\gamma\rangle\in\boldsymbol{\Sigma_1}$,

4. $\mu^{\rightarrow}\in\boldsymbol{\Sigma_0}\ \&\ ₽\in\boldsymbol{D}(\mu)\ \vdash\ \mu^{\rightarrow}\cdot\langle ₽\rangle\in\boldsymbol{\Sigma_1}\ \&\ \mu^{\rightarrow}\cdot\langle ₽,\Delta\rangle\in\boldsymbol{\Sigma_1}$,

5. $\mu^{\rightarrow}\in\boldsymbol{\Sigma_0}\ \&\ u\in\boldsymbol{U}(\mu)\ \vdash\ \mu^{\rightarrow}\cdot\langle u^{\rightarrow}\rangle\in\boldsymbol{\Sigma_0}$.

□

5.6.3. Теорема о пополнении с не-отказами

Далее до конца этого подраздела будем предполагать, что в алфавите $L\subseteq Z$ задана $\mathfrak{R}/\mathfrak{Q}$-семантика и ***F***-спецификация $\boldsymbol{\Sigma}$ с отношением ***safe by***.

Лемма 26: Множество финальных трасс $\boldsymbol{final}(\boldsymbol{\Sigma})$ обладает всеми свойствами $\mathfrak{R}^{\rightarrow}$-модели в алфавите $L^{\Rightarrow}$, кроме, быть может, замкнутости.

□482

Лемма 27: ***d***-замыкание множества финальных трасс является $\mathfrak{R}^{\rightarrow}$-моделью в алфавите $L^{\Rightarrow}$: $\cup\circ\boldsymbol{d}\circ\boldsymbol{final}(\boldsymbol{\Sigma})\in\boldsymbol{MODEL}(L^{\Rightarrow}_{\mathfrak{R}^{\rightarrow}\Delta\gamma})$.

□485

Определение 75: Обозначим $\boldsymbol{Comp}=\boldsymbol{Ext}\circ\cup\circ\boldsymbol{d}\circ\boldsymbol{final}$.

□

Лемма 28: $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ является ***F***-моделью в алфавите $L^{\Rightarrow}$, и её трассы не содержат $\mathfrak{Q}^{\rightarrow}$-отказов.

□486

Определение 76: Определим на ***F***-модели $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ в $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$-семантике отношение ***safe by*** $=_{\text{def}}$ ***safe in***.

□

Лемма 29: 1) Определение ***safe by*** для ***F***-модели ***Comp***(**Σ**) корректно, то есть удовлетворяет всем трём правилам протокола. 2) При таком определении ***safe by*** кнопка "P" опасна после трассы тогда и только тогда, когда не-отказ ₽ разрушающий. 3) Отношение ***safe by*** для ***F***-модели ***Comp***(**Σ**) в $\mathfrak{R}^{\rightarrow}\cup\mathfrak{Q}^{\rightarrow}/\varnothing$-семантике совпадает с его определением в $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$-семантике.

□486

Лемма 30: Если трасса μ безопасна в спецификации **Σ**, то трасса $\mu^{\rightarrow}$ безопасна в спецификации ***Comp***(**Σ**) и, если отказ $P\in\mathfrak{R}\cup\mathfrak{Q}$ безопасен после μ в **Σ**, то отказ $P^{\rightarrow}$ безопасен после $\mu^{\rightarrow}$ в ***Comp***(**Σ**).

□487

Лемма 31: Преобразование ***Comp*** не сужает классы безопасных и конформных реализаций:

$\boldsymbol{safe\mathfrak{I}}(\boldsymbol{\Sigma})\subseteq(\boldsymbol{safe\mathfrak{I}}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma}))^{\leftarrow}$ и $\boldsymbol{\mathfrak{I}}(\boldsymbol{\Sigma})\subseteq(\boldsymbol{\mathfrak{I}}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma}))^{\leftarrow}$.

□488

Лемма 32: Преобразование ***Comp*** не расширяет классы безопасных и конформных реализаций без не-отказов:

$\boldsymbol{safe\mathfrak{I}}(\boldsymbol{\Sigma}) \supseteq (\boldsymbol{safe\mathfrak{I}}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma}))^{\leftarrow}\cap\boldsymbol{FMODEL}(L)$,

$\boldsymbol{\mathfrak{I}}(\boldsymbol{\Sigma}) \supseteq (\boldsymbol{\mathfrak{I}}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma}))^{\leftarrow}\cap\boldsymbol{FMODEL}(L)$.

□493

Лемма 33: Для ***F***-спецификации ***Comp***(**Σ**) в $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$- и $\mathfrak{R}^{\rightarrow}\cup\mathfrak{Q}^{\rightarrow}/\varnothing$-семантиках класс безопасных реализаций вложен, а класс конформных реализаций тот же самый:

$\boldsymbol{safe\mathfrak{I}}_{\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}\circ}\boldsymbol{Comp}(\boldsymbol{\Sigma}) \subseteq \boldsymbol{safe\mathfrak{I}}_{\mathfrak{R}^{\rightarrow}\cup\mathfrak{Q}^{\rightarrow}/\varnothing\circ}\boldsymbol{Comp}(\boldsymbol{\Sigma})$,

$\boldsymbol{\mathfrak{I}}_{\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}\circ}\boldsymbol{Comp}(\boldsymbol{\Sigma}) = \boldsymbol{\mathfrak{I}}_{\mathfrak{R}^{\rightarrow}\cup\mathfrak{Q}^{\rightarrow}/\varnothing\circ}\boldsymbol{Comp}(\boldsymbol{\Sigma})$.

□495

Как итог имеем следующую Теорему.

Теорема 22: Преобразование $\boldsymbol{Comp}(\Sigma)$ является пополнением, сохраняющим класс конформных и не сужающий класс безопасных реализаций, независимо от того, рассматривать результат преобразования в $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$- или в $\mathfrak{R}^{\rightarrow}\cup\mathfrak{Q}^{\rightarrow}/\varnothing$-семантике:

$$\boldsymbol{safe}\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma) = (\ \boldsymbol{safe}\mathfrak{I}_{\mathfrak{R}\rightarrow/\mathfrak{Q}\rightarrow\circ}\boldsymbol{Comp}(\Sigma)\ )^{\leftarrow} \cap \boldsymbol{FMODEL}(\mathtt{L})$$

$$\subseteq (\ \boldsymbol{safe}\mathfrak{I}_{\mathfrak{R}\rightarrow\cup\mathfrak{Q}\rightarrow/\varnothing\circ}\boldsymbol{Comp}(\Sigma)\ )^{\leftarrow} \cap \boldsymbol{FMODEL}(\mathtt{L}),$$

$$\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma) = (\ \mathfrak{I}_{\mathfrak{R}\rightarrow/\mathfrak{Q}\rightarrow\circ}\boldsymbol{Comp}(\Sigma)\ )^{\leftarrow} \cap \boldsymbol{FMODEL}(\mathtt{L})$$

$$= (\ \mathfrak{I}_{\mathfrak{R}\rightarrow\cup\mathfrak{Q}\rightarrow/\varnothing\circ}\boldsymbol{Comp}(\Sigma)\ )^{\leftarrow} \cap \boldsymbol{FMODEL}(\mathtt{L}).$$

□495

5.6.4. Алгоритмизация

Для конечного алфавита преобразование пополнения алгоритмизуемо. Для этого достаточно отметить следующие два факта.

1. Для трассы μ вместо проверки всех трасс из $\boldsymbol{drt}(\mu)$ в определении $\boldsymbol{safe}(\mu)$ и $\mu\ \boldsymbol{conf}$ можно ограничиться только трассами из $\boldsymbol{dt}(\mu)$. Соответствующим образом изменённые определения обозначим $\boldsymbol{G}`(\mu)$, $\boldsymbol{D}`(\mu)$ и $\boldsymbol{U}`(\mu)$.
2. В правиле вывода 5 достаточно продолжать трассу только таким отказом, который ещё не входит в её постфикс. В результате будут строиться финальные трассы без повторных отказов. После нужно делать не ***d***-, а ***dr***-замыкание.

Для LTS-спецификации $\mathbf{s}$ обозначения из Определения 73 понимаются для $\Sigma=\boldsymbol{F}(\mathbf{s})$. В LTS-пополнении в качестве состояний можно выбрать пустую трассу (начальное состояние), γ-состояние (в нём определяется γ-петля), Δ-состояние (в нём определяется τ-петля) и состояния, соответствующие трассам из множества Σ_0 трассового пополнения, которые либо a) не заканчиваются на отказ, либо b) заканчиваются непустым постфиксом отказов без повторных

отказов. Из состояний вида a) проводятся τ-переходы во все продолжения вида b).

Обозначим:

$\boldsymbol{Z}`(\mu) =_{\text{def}} \boldsymbol{U}`(\mu) \cap \mathtt{L}$,

$\boldsymbol{R}`(\mu) =_{\text{def}} \{\ \rho \in \mathfrak{R}^{*} \ \mid \ |\rho| = |\boldsymbol{Im}(\rho)| > 0$

$\qquad\qquad \&\ \forall i \in [1..|\rho|]\ \rho(i) \in \boldsymbol{U}`(\mu \cdot \rho[1..i-1])\ \}.$

Для каждой трассы μ, которая не содержит разрушение, дивергенцию и не-отказы, определяем:

1. $s_0 = \langle\gamma\rangle \Rightarrow \quad \vdash \epsilon \text{—}\gamma\text{→} \epsilon,$

2. $s_0 = \langle\gamma\rangle \not\Rightarrow \ \&\ P \in \boldsymbol{G}`(\mu) \quad \vdash \mu^{\rightarrow} \text{—}\not{\tau}\text{→} \gamma \text{—}\gamma\text{→} \gamma,$

3. $s_0 = \langle\gamma\rangle \not\Rightarrow \ \&\ P \in \boldsymbol{D}`(\mu) \quad \vdash \mu^{\rightarrow} \text{—}\not{\tau}\text{→} \Delta \text{—}\tau\text{→} \Delta,$

4. $s_0 = \langle\gamma\rangle \not\Rightarrow \ \&\ z \in \boldsymbol{Z}`(\mu) \quad \vdash \mu^{\rightarrow} \text{—}z\text{→} \mu^{\rightarrow} \cdot \langle z \rangle,$

5. $s_0 = \langle\gamma\rangle \not\Rightarrow \ \&\ \boldsymbol{postf}(\mu) = \epsilon \ \&\ \rho \in \boldsymbol{R}`(\mu) \quad \vdash \mu^{\rightarrow} \text{—}\tau\text{→} \mu^{\rightarrow} \cdot \rho^{\rightarrow}.$

Отметим, что при конечном алфавите множество $\boldsymbol{R}`(\mu)$ конечно, и поэтому «веер» τ-переходов из состояния $\mu^{\rightarrow}$, порождаемый последним правилом вывода, также конечен, что требуется для алгоритмического задания преобразованной LTS. В случае бесконечного алфавита вопросы разработки алгоритмов пополнения более тонкие. Здесь мы не будем в них углубляться, поскольку это выходит за рамки темы данной работы.

5.6.5. Частные случаи δ-, γδ- и βγδ-семантик

В этих семантиках отношение ***safe by*** определяется однозначно, поскольку кнопочные множества не пересекаются.

Как было отмечено выше (5.1), в δ-семантике (семантике для отношения ***ioco***) существует проблема выхода за пределы домена отношения ***ioco*** при композиции: композиция строго-конвергентных LTS может иметь достижимую

дивергенцию, возникающую как бесконечная последовательность синхронных переходов. Поэтому предпочтительнее рассматривать более общий случай γδ-семантики, допускающей как дивергенцию, так и разрушение.

В γδ-семантике $\mathfrak{Q}$-отказами являются блокировки стимулов, а единственный $\mathfrak{R}$-отказ – это δ = множество всех реакций. Поскольку кнопочные множества не пересекаются, для проверки безопасности стимула `?x` или реакции после трассы достаточно проверять безопасность только одной кнопки: блокировки стимула `{?x}` или δ, соответственно. В остальном преобразование пополнения такое же, как в общем случае.

В βγδ-семантике нет $\mathfrak{Q}$-кнопок и поэтому не требуется пополнение.

5.7. Выводы

В этой главе мы изучали проблему пополнения спецификации. Задачей такого пополнения является достижение рефлексивности конформности, что, в свою очередь, является первым шагом к решению проблемы монотонности (сохранения конформности при композиции).

Мы рассмотрели проблему пополнения для частного случая δ-семантики для реактивных систем и отношения ***ioco***, отличающегося от отношения $\boldsymbol{ioco}_\delta$ (***saco*** в δ-семантике) только более узким доменом реализаций и спецификаций. Показано, что отношение ***ioco***, вообще говоря, нерефлексивно, транзитивно и немонотонно. Также для δ-семантики существует проблема выхода за пределы домена отношения ***ioco*** (и $\boldsymbol{ioco}_\delta$) при композиции (возникновение дивергенции как бесконечной цепочки синхронных переходов). Эта проблема решается в предлагаемом отношении $\boldsymbol{ioco}_{\gamma\delta}$ (***saco*** в γδ-семантике). Рассмотрены различные виды допущений полноты, используемые в литературе по тестированию конформности и предлагается новое гамма-пополнение. Показано, что отношение ***ioco***, вообще говоря, не сохраняется при пополнении состояний LTS-модели, причём только два вида пополнения состояний не

усиливают конформность: гамма-пополнения и демоническое пополнение. При этом гамма-пополнение удобнее тем, что сохраняется информация о стимулах, которыми не продолжались трассы: теперь эти стимулы принимаются с дальнейшим разрушением, в то время как при демоническом пополнении эта информация теряется. Тем самым, при гамма-пополнении мы избегаем лишнего тестирования, которое неизбежно возникает при демоническом пополнении. В целом делается вывод о необходимости трассового гамма-пополнения.

Далее исследовался вопрос о рефлексивности и транзитивности отношения и было сформулировано необходимое и достаточное условие рефлексивности и достаточное условие транзитивности. Показано, что при отсутствии $\mathfrak{Q}$-кнопок эти условия выполнены и в $\mathfrak{R}/\varnothing$-семантике отношение ***saco*** является предпорядком.

Наконец, было доказано существование пополнения, сохраняющего класс конформных реализаций и такого, что для пополненной спецификации классы конформных реализаций в $\mathfrak{R}/\mathfrak{Q}$- и $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантиках совпадают. Это пополнение строится как объединение конформных трассовых реализаций.

После этого изучалась проблема изменения класса безопасных реализаций. Показано, что этот класс может как расширяться (что допустимо и даже хорошо), так и сужаться (что плохо) при пополнении спецификации. Исследовались причины сужения класса безопасных реализаций и было предложено решение проблемы, основанное на введении фиктивных действий – не-отказов. С помощью не-отказов удаётся сделать такое пополнение, которое не только сохраняет конформные реализации, но и не сужает класс безопасных реализаций. Кратко были рассмотрены вопросы алгоритмизации этого пополнения и его частные случаи для δ-, γδ- и βγδ-семантик.

Новыми результатами являются:

1) Метод гамма-пополнения состояний LTS-спецификации в семантике отношения ***ioco*** (для систем ввода-вывода без блокировок стимулов, но с возможностью наблюдения отсутствия реакций). Этот метод отличается от

известного демонического пополнения [66] тем, что не определённый в состоянии спецификации стимул заменяется на приём этого стимула не с произвольным дальнейшим поведением, а с единственным дальнейшим действием – разрушением. Оба эти пополнения выгодно отличаются от остальных предлагаемых в литературе пополнений состояний тем, что не усиливают конформность и, следовательно, тесты, сгенерированные по таким пополненным спецификациям не ловят ложных ошибок. Однако гамма-пополнение имеет то преимущество перед демоническим пополнением, что не теряется информация о стимуле, который был не определён в исходной спецификации. Это даёт возможность избегать лишних тестов при генерации полного тестового набора по пополненной спецификации, в частности, при переходе от синхронного к асинхронному тестированию.

2) Доказательство существования пополнения спецификаций, не изменяющего (не только не усиливающего, но и не ослабляющего) конформность и основанного на объединении конформных реализаций (как моделей наблюдаемых трасс). Такое пополнение позволяет перейти к семантике, в которой все отказы наблюдаемы, и которая является отношением предпорядка (в частности, рефлексивна).
3) Метод пополнения спецификаций с использованием специальных фиктивных действий «не-отказов». Во-первых, этот метод, также как объединение конформных реализаций, позволяет перейти к семантике, в которой все отказы наблюдаемы и которая является отношением предпорядка. Во-вторых, этот метод, в отличие от объединения конформных реализаций, позволяет не только сохранить при пополнении класс конформных, но и не сузить класс безопасных реализаций. И, в-третьих, этот метод поддаётся алгоритмизации при некоторых ограничениях на спецификацию. Частное применение этого метода решает проблему пополнения для семантики отношения ***ioco***.

Глава 6. Верификация композиции

Структура главы:

1. Общая теория монотонности
2. $\mathfrak{R}$-мажорирование и конформность
3. ϕ-трассы
4. Объединение ϕ-трасс конформных реализаций
5. Мажорирование ϕ-трасс
6. Монотонные модели
7. Спецификации с ограниченной ветвимостью
8. Монотонное преобразование
9. Композиция
10. Выводы

В этой главе речь пойдёт о проблеме монотонности, напомним вкратце суть этой проблемы.

Композиционная система – это составная система, собранная из компонентов с помощью применения определенных правил композиции. В этой работе компоненты моделируются LTS, а правила композиции – оператором $\uparrow\downarrow$ параллельной композиции LTS (Определение 56). Мы допускаем θ-переходы только в тесте, но не в реализации. Поэтому компоненты системы моделируются LTS без θ-переходов, а их композиция – оператором $\uparrow\downarrow$ без правил вывода $(\uparrow\downarrow 4,5)$.

Основная проблема композиционных систем звучит так: если компоненты работают правильно, то почему система в целом работает неправильно? В теории конформности правильность реализации отдельного компонента определяется как конформность реализации компонента спецификации

компонента, а правильность реализации системы – как конформность реализации системы спецификации системы.

Правильность реализации компонента проверяется тестированием этого компонента, когда тест непосредственно взаимодействует с компонентом, подменяя собой его окружение. Такое тестирование называют *автономным* (тестируется отдельный компонент) или *синхронным*. 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 проблема композиционных систем этим не исчерпывается. Реализации компонентов могут быть конформны своим спецификациям, а собранная из этих компонентов система неконформна 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 [27,43,66,67]. 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 компонентов, конформных своим спецификациям, конформна спецификации системы. Спецификация системы *корректна*, если она удовлетворяет условию монотонности при заданных спецификациях компонентов. Самая сильная (то

есть, предъявляющая максимальные 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.

Проблема в том, что самая сильная корректная спецификация системы, если она существует, оказывается слабее композиции спецификаций, проводимой по тем же правилам, что и композиция реализаций компонентов. Если композицию реализаций $\between$ 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 конформности и прямой композиции. Конформность основана на трассах наблюдений над поведением реализационной модели, а композиция, кроме того, на ненаблюдаемых напрямую состояниях. Для отношения $\boldsymbol{saco}$ наблюдения — это внешние (наблюдаемые) действия и $\mathfrak{R}$-отказы (наблюдаемое отсутствие выполнения действий). Композиция дополнительно учитывает состояния, ненаблюдаемые действия (τ-действия) и соотношение состояний и действий.

Особым случаем композиции является асинхронное тестирование или тестирование в контексте [141]. Такое тестирование можно рассматривать как тестирование системы из двух компонентов, один из которых – реализация, а другой – фиксированная среда взаимодействия. При асинхронном тестировании, в отличие от синхронного тестирования, возникают две проблемы [105]: 1) «вседозволенность» (*permissiveness*), когда асинхронные тесты не ловят ошибку, обнаруживаемую синхронными тестами, и 2) «несохранение соответствия» (*non preservation of conformance*), когда асинхронные тесты ловят «ложную» ошибку. С первой проблемой, по-видимому, приходится мириться: асинхронное (и вообще композиционное) тестирование – это более «косвенное» тестирование реализации. Оно может не позволить создать все те режимы взаимодействия и наблюдать всё то поведение

реализации, которые возможны в синхронном тестировании, когда тест и реализация взаимодействуют непосредственно друг с другом. Вторая проблема более серьёзная – это частный случай общей проблемы монотонности.

Таким образом, ставится задача построения косой композиции для выбранной конформности. Пополнение спецификаций – это первый шаг к решению этой задачи. У нас появляется возможность решать эту задачу только для $\mathfrak{R}/\varnothing$-семантики, когда $\mathfrak{Q}$-кнопок нет. В дальнейшем под спецификацией будем понимать спецификацию, для которой уже выполнено необходимое пополнение, и совершён переход от $\mathfrak{R}/\mathfrak{Q}$-семантики к $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-семантике. Напомним, что в семантике без $\mathfrak{Q}$-кнопок отношения ***safe by*** и ***safe in*** совпадают, и кнопка опасна тогда и только тогда, когда она разрешает разрушающее действие.

Прежде всего, отметим, что пополнением проблема монотонности не исчерпывается. На Рис.61 приведён пример несохранения конформности при асинхронном тестировании для семантики, в которой наблюдаемы все отказы: стационарность и блокировки стимулов. Среда $\mathbf{Q}$ – это одна ограниченная выходная очередь (на рисунке длины 1) с дополнительной командой «обнуления» (b). Спецификация $\mathbf{S}_0$ описывает следующие требования к системе: сначала стимул блокируется, но можно выдавать цепочку реакций a, потом можно обнулить очередь (b), принять стимул x и закончить в терминальном состоянии. Реализация $\mathbf{S}_1$ конформна: она не обнуляет очередь и, соответственно, не принимает стимул. Когда с очередью компонуется спецификация, первый посылаемый стимул не блокируется. Однако при композиции реализации $\mathbf{S}_1$ такая блокировка появляется, что при асинхронном тестировании будет квалифицировано как ошибка (показано красными стрелками).

Несохранение конформности при асинхронном тестировании для семантики, в которой все отказы наблюдаемы

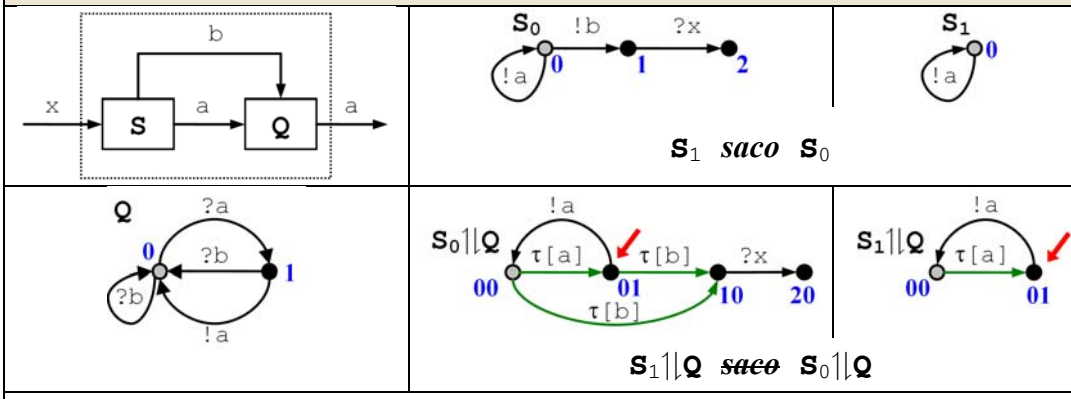


Рис.61.

В данной главе даётся формальное определение косой композиции для отношения $\boldsymbol{saco}_{\mathfrak{R}/\varnothing}$. Будет показано, что косая композиция совпадает с прямой композицией *преобразованных* спецификаций. Такое преобразование будем называть монотонным для этой конформности, а конформность – монотонной относительно этого преобразования.

Следует отметить, что монотонное преобразование определяется неоднозначно. Мы опишем два таких преобразования. Одно основано на объединении конформных реализаций и применимо всегда. Другое применимо лишь при определённых условиях, но зато определяется конструктивно (индуктивно), что даёт возможность его алгоритмизации.

Заметим также, что разные компоненты могут иметь разные алфавиты, но даже в одном алфавите их конформность может быть определена в разных $\mathfrak{R}/\varnothing$-семантиках, то есть при разных наборах $\mathfrak{R}$-кнопок машины тестирования. Возникает вопрос: в какой семантике следует рассматривать композицию компонентов? По счастью, этот вопрос снимается тем, что прямая композиция монотонно преобразованных спецификаций оказывается косой композицией *в любой* $\mathfrak{R}/\varnothing$-семантике (естественно, для одного и того же композиционного алфавита, однозначно определяемого алфавитами операндов и не зависящего от $\mathfrak{R}$-семантик операндов при тех же алфавитах).

При асинхронном тестировании предполагается, что в среде нет ошибок и она известна. Поэтому 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. При решении проблемы монотонности мы будем учитывать и этот случай.

Косая композиция спецификаций компонентов составной системы позволяет решить две задачи: 1) верификация имеющейся спецификации системы (её согласованности со спецификациями компонентов), и 2) при отсутствии спецификации системы её генерация по спецификациям компонентов.

Первая задача иногда называется задачей верификации декомпозиции системных требований, то есть проверкой правильности декомпозиции требований к системе в требования к компонентам системы. Для её решения необходимо построить косую композицию системы и, рассматривая её как реализацию, проверить её конформность имеющейся спецификации системы. Такая проверка может выполняться аналитически, в том числе, моделируя тестирование результата косой композиции, рассматриваемой как реализация системы, по полному набору тестов, сгенерированному из заданной спецификации системы.

В качестве примера можно рассмотреть композицию (Рис.62) приёмника с двумя входными портами (Рис.19) и передатчика с двумя выходными портами (Рис.20). Спецификация составной системы очень проста: нужно принять пару аргументов (x,y), и выдать значение $\boldsymbol{f}(\boldsymbol{g}(x),\boldsymbol{h}(y))$. Интуитивно именно это и должны делать в совокупности реализации компонентов: передатчик $\mathbf{T}$ со спецификаций $\mathbf{T}_0$ вычисляет функции $\boldsymbol{g}$ и $\boldsymbol{h}$, а приёмник $\mathbf{F}$ со спецификаций $\mathbf{F}_0$ вычисляет функцию $\boldsymbol{f}$. Каждый компонент рассматривается в двух семантиках: в одной семантике конформна только сама спецификация, в другой – все реализации. Зелёные клетки соответствуют случаям, когда

композиция реализаций компонентов конформна спецификации системы, а жёлтые клетки – когда это не так: в композиции реализаций реакции может не быть. Мы видим, что не всякое сочетание семантик компонентов **T** и **F** гарантирует согласованность спецификаций компонентов со спецификацией системы. Из четырёх вариантов, одно ошибочное, а три правильных. В главе 2 мы расматривали только одно правильное сочетание: передатчик передаёт сразу два сообщения, а приёмник принимает их в любом порядке.

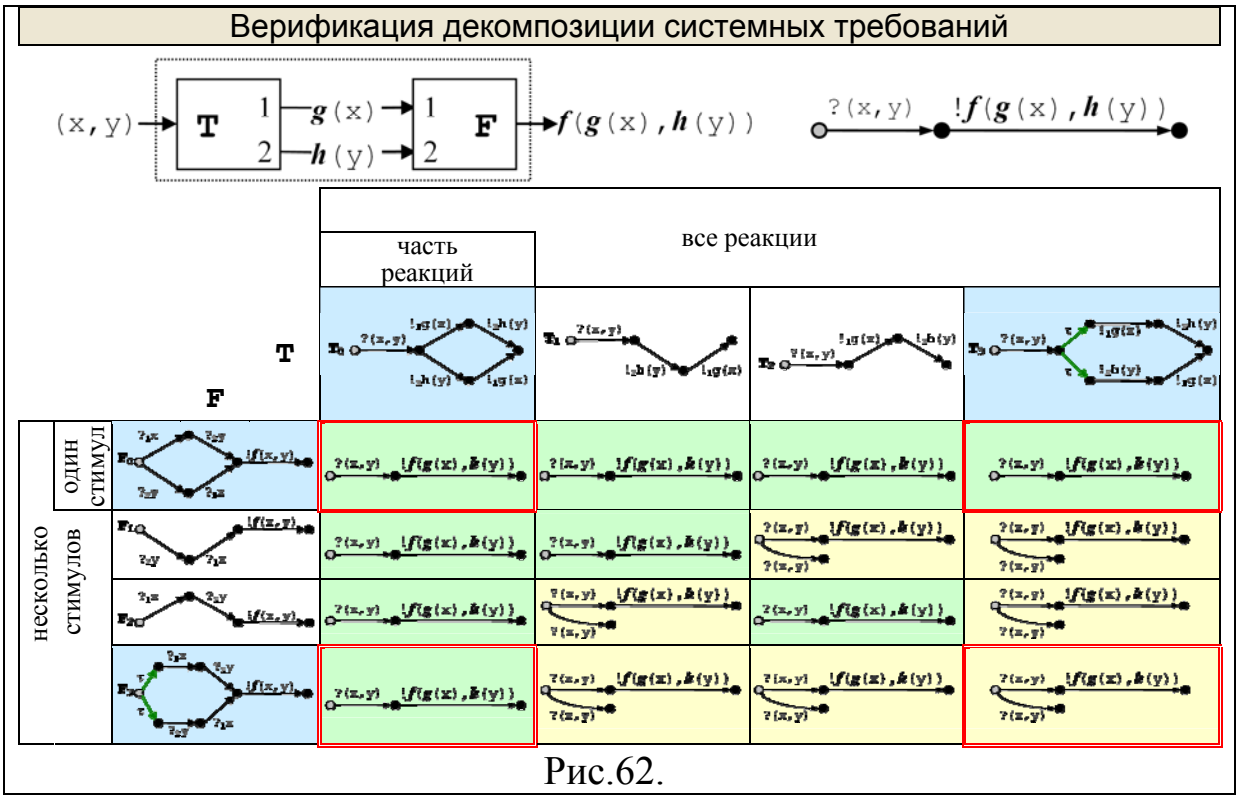


Рис.62.

Как же верифицировать декомпозицию? На рисунке для каждого компонента синим фоном клетки показаны монотонно преобразованные спецификации в зависимости от семантики, в которой рассматривалась исходная спецификация. Красным обведены те клетки таблицы, в которых нарисована как раз композиция монотонно преобразованных спецификаций, но только в сокращённом виде, без τ-переходов. В зелёных клетках получается композиция спецификаций, которая конформна заданной спецификации, а в одной жёлтой клетке – неконформная композиция спецификаций. В этом последнем случае спецификации компонентов и системы не согласованы.

Решение второй задачи (генерация спецификации системы) преследует три цели. 1) Полученная спецификация системы используется как документ, описывающий систему с точки зрения её пользователей. 2) Такая спецификация может рассматриваться как техническое задание разработчику системы, в том числе, при возможных дальнейших модификациях системы (её компонентов или даже самой схемы композиции). 3) По спецификации системы могут генерироваться системные (комплексные) тесты.

Остановимся немного подробнее на последней цели. Теоретически, если компоненты полностью протестированы на конформность своим спецификациям, нам нет нужды тестировать саму систему. Вместо этого достаточно либо убедиться в правильном соотношении имеющихся спецификаций системы и её компонентов (первая задача), либо просто сгенерировать отсутствующую спецификацию системы по имеющимся спецификациям компонентов (первые две цели второй задачи). Однако, в связи с тем, что полный тестовый набор, как правило, бесконечен, всякое практическое тестирование оказывается не полным[24] и, следовательно, мы не можем быть стопроцентно уверены, что автономное тестирование выловило все ошибки в компонентах. Именно поэтому на практике всегда остаётся нужда в системных тестах. В частности, системное тестирование создаёт такие режимы взаимодействия компонентов, которые, с одной стороны, ближе к режимам реальной эксплуатации системы, а, с другой стороны, часто не воспроизводятся в практическом автономном тестировании (или это очень трудно сделать).

С системным тестированием связаны две проблемы. Во-первых, косая композиция спецификаций должна удовлетворять тем требованиям, которые делают возможным алгоритмическую генерацию тестов. Для этого приходится налагать на исходные спецификации дополнительные ограничения. Во-вторых,

---

[24] Тестирование может считаться полным, если на класс тестируемых реализаций налагаются специальные, обычно довольно сильные, ограничения. Например, при тестировании конечных автоматов считают, что число состояний реализации не превосходит числа состояний спецификации или превосходит не более, чем на фиксированное число [30,56,112,113].

системное тестирование должно быть безопасным. Как будет показано в этой главе, без дополнительных гипотез о возможных реализациях компонентов, кроме их безопасности, или специальных средств проверки безопасности композиция реализаций компонентов может оказаться опасной для практически любой спецификации системы, если такие компоненты вообще взаимодействуют друг с другом. В результате системное тестирование либо опасно, либо сводится к автономному тестированию отдельных компонентов, которые не взаимодействуют друг с другом.

В теории монотонности нам придётся иметь дело с конформностью и композицией. Конформность определяется в рамках теории трасс наблюдений, а композиция определяется для LTS-моделей. При этом трасс наблюдений ($\mathfrak{R}$-трасс) недостаточно для определения композиции: одной модели трасс наблюдений соответствует множество различных LTS, в том числе LTS, дающих при композиции существенно разные результаты с точки зрения конформности.

Тем не менее, в трассовой теории также можно определить композицию моделей, имеющую тот же смысл, если использовать другие трассы. Мы вводим трассы, которые будем называть ϕ-трассами, и определим композицию ϕ-трасс таким образом, что ϕ-трассы композиции LTS-моделей совпадают с композицией ϕ-трасс этих моделей. Это свойство мы будем называть *аддитивностью* ϕ-трасс относительно композиции. Заметим, что ϕ-трассы не являются, вообще говоря, трассами наблюдений, по крайней мере, в $\mathfrak{R}/\mathfrak{Q}$-семантике. Вместе с тем, по ϕ-трассам LTS-модели можно получить все её $\mathfrak{R}$-трассы, хотя обратное неверно. Это свойство мы будем называть *генеративностью* ϕ-трасс.

ϕ-трассы похожи на трассы готовности (*ready traces*), но имеют два отличия. 1) Вместо множества готовности (*ready set*) мы используем его дополнение – ***ref***-множество, то есть множество действий, которые не могут выполняться в данном стабильном состоянии. Это сделано по аналогии с

отвергаемым множеством (*refusal set*) в трассах с отказами (*failure traces*) и не носит принципиального характера. 2) Также мы вводим ***gamma***-множество: множество (непосредственно) разрушающих действий, определённых в данном стабильном состоянии. Это сделано для удобства построения теории. Пару (***ref***-множество, ***gamma***-множество) мы называем ϕ-символом, и ϕ-трасса определяется как последовательность внешних действий и ϕ-символов. Понятно, что по ϕ-трассам LTS-модели легко вычисляются её трассы готовности.

6.1. Общая теория монотонности

Структура раздела:

1. Корректная спецификация, косая композиция и монотонность
2. Восемь достаточных условий монотонности

В этом разделе излагается формальная теория монотонности в самых общих терминах модели и конформности, являющейся предпорядком на множестве моделей. В подразделе 6.1.1 формально определяются понятия корректной спецификации композиции, косой композиции и монотонности. В подразделе 6.1.2 формулируются восемь условий монотонности и доказывается их достаточность.

6.1.1. Корректная спецификация, косая композиция и монотонность

Поскольку в этой главе мы будем рассматривать только семантики без $\mathfrak{Q}$-кнопок, для краткости вместо $\mathfrak{R}/\varnothing$ будем писать просто $\mathfrak{R}$. Если не оговорено противное, мы будем предполагать, что в алфавите $L \subseteq Z^{\#}$ задана $\mathfrak{R}$-семантика: $\cup\mathfrak{R}=L$. Поскольку в $\mathfrak{R}$-семантике отношения ***safe by*** и ***safe in***

совпадают, мы будем писать просто ***safe***. Также вместо ***SafeBy*** и ***SafeIn*** мы будем писать ***Safe***.

Напомним, что, согласно Теорема 17:, в любой $\mathfrak{R}$-семантике отношение $saco_{\mathfrak{R}}$ является предпорядком (рефлексивно и транзитивно). В литературе понятия верхнего/нижнего конуса и точной верхней/нижней грани подмножества обычно определяются для частичного порядка, определённого на множестве [47]. Аналогично можно определить эти понятия для класса, на котором задано отношение предпорядка. В нашем случае таким классом является класс LTS в одном алфавите L, а предпорядком – отношение $saco_{\mathfrak{R}}$ для соответствующей этому алфавиту $\mathfrak{R}$-семантики (то есть $\cup\mathfrak{R}=L$).

Определение 77: Для $S, S' \in LTS(L_\gamma)$ и $\mathfrak{C}(L_\gamma) \subseteq LTS(L_\gamma)$ обозначим:

- верхний (нижний) конус подкласса – класс элементов больше или равных (меньше или равных) каждому элементу подкласса:

  $$\mathfrak{C}(L_\gamma)^{\triangle}_{\mathfrak{R}} =_{def} \{S_1 \in LTS(L_\gamma) \mid \forall S \in \mathfrak{C}(L_\gamma)\ S\ saco_{\mathfrak{R}}\ S_1\},$$

  $$\mathfrak{C}(L_\gamma)^{\triangledown}_{\mathfrak{R}} =_{def} \{S_1 \in LTS(L_\gamma) \mid \forall S \in \mathfrak{C}(L_\gamma)\ S_1\ saco_{\mathfrak{R}}\ S\};$$

- точная верхняя (нижняя) грань подкласса – класс наименьших (наибольших) элементов верхнего (нижнего) конуса подкласса:

  $$sup_{\mathfrak{R}}(\mathfrak{C}(L_\gamma)) =_{def} \{S_2 \in \mathfrak{C}(L_\gamma)^{\triangle}_{\mathfrak{R}} \mid \forall S_1 \in \mathfrak{C}(L_\gamma)^{\triangle}_{\mathfrak{R}}\ S_2\ saco_{\mathfrak{R}}\ S_1\},$$

  $$inf_{\mathfrak{R}}(\mathfrak{C}(L_\gamma)) =_{def} \{S_2 \in \mathfrak{C}(L_\gamma)^{\triangledown}_{\mathfrak{R}} \mid \forall S_1 \in \mathfrak{C}(L_\gamma)^{\triangledown}_{\mathfrak{R}}\ S_1\ saco_{\mathfrak{R}}\ S_2\}.$$

□

Очевидно, что класс всех реализаций, конформных по отношению $saco_{\mathfrak{R}}$ данной спецификации S, является её верхним конусом: $\mathfrak{I}_{\mathfrak{R}}(S) = \{S\}^{\triangledown}_{\mathfrak{R}}$.

Определение 78: Корректная спецификация и косая композиция. Пусть в алфавитах L_1, L_2 и $L=L_1 \wr\!\downarrow L_2$ заданы $\mathfrak{R}_1$-, $\mathfrak{R}_2$- и $\mathfrak{R}$-семантики, соответственно: $\cup\mathfrak{R}_1=L_1$, $\cup\mathfrak{R}_2=L_2$, $\cup\mathfrak{R}=L$.

· Для спецификации $\mathbf{S}_1 \in \boldsymbol{LTS}(L_{1\gamma})$ и реализации $\mathbf{I}_2 \in \boldsymbol{LTS}(L_{2\gamma})$ $\mathfrak{R}$-*левокорректной* спецификацией их композиции назовём спецификацию $\mathbf{S}$, которой конформна по отношению $\boldsymbol{saco}_{\mathfrak{R}}$ композиция любой реализации, конформной по отношению $\boldsymbol{saco}_{\mathfrak{R}1}$ спецификации $\mathbf{S}_1$, с реализацией $\mathbf{I}_2$. Если спецификация $\mathbf{S}$ $\mathfrak{R}$-левокорректна для любой $\mathfrak{R}$-семантики, где $\cup\mathfrak{R}=L$, то будем называть её просто левокорректной спецификацией. Классы $\mathfrak{R}$-левокорректных и левокорректных спецификаций обозначим:

$$\mathfrak{S}_{\mathfrak{R}1,\mathfrak{R}}(\mathbf{S}_1)(\mathbf{I}_2) =_{\text{def}} (\mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\downarrow \mathbf{I}_2)^{\triangle}{}_{\mathfrak{R}},$$

$$\mathfrak{S}_{\mathfrak{R}1}(\mathbf{S}_1)(\mathbf{I}_2) =_{\text{def}} \{\mathbf{S} \mid \forall\mathfrak{R}\ \cup\mathfrak{R}=L \Rightarrow \mathbf{S} \in \mathfrak{S}_{\mathfrak{R}1,\mathfrak{R}}(\mathbf{S}_1)(\mathbf{I}_2)\}.$$

· Для двух спецификаций $\mathbf{S}_1 \in \boldsymbol{LTS}(L_{1\gamma})$ и $\mathbf{S}_2 \in \boldsymbol{LTS}(L_{2\gamma})$ $\mathfrak{R}$-*корректной* спецификацией их композиции назовём спецификацию $\mathbf{S}$, которой конформна по отношению $\boldsymbol{saco}_{\mathfrak{R}}$ композиция любых реализацией, конформных по отношениям $\boldsymbol{saco}_{\mathfrak{R}1}$ и $\boldsymbol{saco}_{\mathfrak{R}2}$ спецификациям $\mathbf{S}_1$ и $\mathbf{S}_2$, соответственно. Если спецификация $\mathbf{S}$ $\mathfrak{R}$-корректна для любой $\mathfrak{R}$-семантики, где $\cup\mathfrak{R}=L_1 \wr\!\downarrow L_2$, то будем называть её просто корректной спецификацией. Классы $\mathfrak{R}$-корректных и корректных спецификаций обозначим:

$$\mathfrak{S}_{\mathfrak{R}1,\mathfrak{R}2,\mathfrak{R}}(\mathbf{S}_1,\mathbf{S}_2) =_{\text{def}} (\mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\downarrow \mathfrak{I}_{\mathfrak{R}2}(\mathbf{S}_2))^{\triangle}{}_{\mathfrak{R}},$$

$$\mathfrak{S}_{\mathfrak{R}1,\mathfrak{R}2}(\mathbf{S}_1,\mathbf{S}_2) =_{\text{def}} \{\mathbf{S} \mid \forall\mathfrak{R}\ \cup\mathfrak{R}=L \Rightarrow \mathbf{S} \in \mathfrak{S}_{\mathfrak{R}1,\mathfrak{R}2,\mathfrak{R}}(\mathbf{S}_1,\mathbf{S}_2)\}.$$

· $\mathfrak{R}$-*левокосой композицией* спецификации $\mathbf{S_1}\in\boldsymbol{LTS}(\mathtt{L}_{1\gamma})$ и реализации $\mathbf{I_2}\in\boldsymbol{LTS}(\mathtt{L}_{2\gamma})$ будем называть самую сильную $\mathfrak{R}$-левокорректную спецификацию их композиции. Если спецификация $\mathbf{S}$ является $\mathfrak{R}$-левокосой композицией для любой $\mathfrak{R}$-семантики, где $\cup\mathfrak{R}=\mathtt{L}$, то будем называть её просто левокосой композицией. Классы $\mathfrak{R}$-левокосых и левокосых композиций обозначим:

$\mathbf{S_1}\not\downarrow_{\mathfrak{R}1,\mathfrak{R}}\downarrow\mathbf{I_2} =_{\text{def}} \boldsymbol{sup}_{\mathfrak{R}}(\mathfrak{I}_{\mathfrak{R}1}(\mathbf{S_1})\,\uparrow\!|\!\downarrow\mathbf{I_2})$,

$\mathbf{S_1}\not\downarrow_{\mathfrak{R}1}\downarrow\mathbf{I_2} =_{\text{def}} \{\mathbf{S}\,|\,\forall\mathfrak{R}\ \cup\mathfrak{R}=\mathtt{L}\ \Rightarrow\ \mathbf{S}\in\mathbf{S_1}\not\downarrow_{\mathfrak{R}1,\mathfrak{R}}\downarrow\mathbf{I_2}\}$.

· $\mathfrak{R}$-*косой композицией* двух спецификаций компонентов $\mathbf{S_1}\in\boldsymbol{LTS}(\mathtt{L}_{1\gamma})$ и $\mathbf{S_2}\in\boldsymbol{LTS}(\mathtt{L}_{2\gamma})$ будем называть самую сильную $\mathfrak{R}$-корректную спецификацию их композиции. Если спецификация $\mathbf{S}$ является $\mathfrak{R}$-косой композицией для любой $\mathfrak{R}$-семантики, где $\cup\mathfrak{R}=\mathtt{L}$, то будем называть её просто косой композицией. Классы $\mathfrak{R}$-косых и косых композиций обозначим:

$\mathbf{S_1}\not\downarrow_{\mathfrak{R}1,\mathfrak{R}2,\mathfrak{R}}\not\downarrow\mathbf{S_2} =_{\text{def}} \boldsymbol{sup}_{\mathfrak{R}}(\mathfrak{I}_{\mathfrak{R}1}(\mathbf{S_1})\,\uparrow\!|\!\downarrow\mathfrak{I}_{\mathfrak{R}2}(\mathbf{S_2}))$,

$\mathbf{S_1}\not\downarrow_{\mathfrak{R}1,\mathfrak{R}2}\not\downarrow\mathbf{S_2} =_{\text{def}} \{\mathbf{S}\,|\,\forall\mathfrak{R}\ \cup\mathfrak{R}=\mathtt{L}\ \Rightarrow\ \mathbf{S}\in\mathbf{S_1}\not\downarrow_{\mathfrak{R}1,\mathfrak{R}2,\mathfrak{R}}\not\downarrow\mathbf{S_2}\}$.

□

Заметим, что мы могли бы аналогично определить ($\mathfrak{R}$-)правокорректную спецификацию и ($\mathfrak{R}$-)правокосую композицию. Однако, в силу коммутативности оператора прямой композиции $\uparrow\!|\!\downarrow$, это излишне. То же самое относится к понятию монотонности, которое мы определяем ниже.

Определение 79: Монотонность. Будем считать, что для каждого алфавита $\mathtt{L}$ выделен подкласс спецификаций $\mathfrak{C}(\mathtt{L}_\gamma)\subseteq\boldsymbol{LTS}(\mathtt{L}_\gamma)$ и задано семейство

преобразований $\mathcal{T}(L)=\{\mathcal{T}_{\mathfrak{R}}:\mathfrak{C}(L_\gamma)\to\mathfrak{C}(L_\gamma)\mid\cup\mathfrak{R}=L\}$. Обозначим:

$\mathfrak{C}=\{\mathbf{S}\mid\mathbf{S}\in\mathfrak{C}(L_\gamma)\ \&\ L\subseteq Z^{\#}\}$ и $\mathcal{T}=\{\mathcal{T}_{\mathfrak{R}}\mid\mathcal{T}_{\mathfrak{R}}\in\mathcal{T}(L)\ \&\ L\subseteq Z^{\#}\}$.

- Будем говорить, что семейство преобразований $\mathcal{T}$ *инвариантно* (на классе $\mathfrak{C}$), если каждое его преобразование $\mathcal{T}_{\mathfrak{R}}$ сохраняет эквивалентность $\sim_{\mathfrak{R}}$ по конформности в $\mathfrak{R}$-семантике:

  $\forall L\subseteq Z^{\#}\ \forall\mathfrak{R}\ \forall\mathbf{S}\in\mathfrak{C}(L_\gamma)\ \cup\mathfrak{R}=L\ \Rightarrow\ \mathcal{T}_{\mathfrak{R}}(\mathbf{S})\ \sim_{\mathfrak{R}}\ \mathbf{S}$.[25]

- Будем говорить, что отношение ***saco*** $\mathcal{T}$*-левомонотонно* (на классе $\mathfrak{C}$), если семейство преобразований $\mathcal{T}$ инвариантно и прямая композиция преобразованной спецификации из $\mathfrak{C}$ с любой реализацией из $\mathfrak{C}$ является левокосой композицией:

  $\forall L_1\subseteq Z^{\#}\ \forall\mathfrak{R}_1\ \forall\mathbf{S}_1\in\mathfrak{C}(L_{1\gamma})\ \forall\mathbf{I}_2\in\mathfrak{C}$

  $\cup\mathfrak{R}_1=L_1\ \Rightarrow\ \mathcal{T}_{\mathfrak{R}_1}(\mathbf{S}_1)\uparrow\downarrow\mathbf{I}_2\in\mathbf{S}_1\not\downarrow_{\mathfrak{R}_1}\downarrow\mathbf{I}_2$.

  В этом случае семейство преобразований $\mathcal{T}$ будем называть левомонотонным (на классе $\mathfrak{C}$), а результат преобразования $\mathcal{T}_{\mathfrak{R}}(\mathbf{S})$ – левомонотонной спецификацией.

- Будем говорить, что семейство отношений ***saco*** $\mathcal{T}$*-монотонно* (на классе $\mathfrak{C}$), если семейство преобразований $\mathcal{T}$ инвариантно и прямая композиция преобразованных спецификаций из класса $\mathfrak{C}$ является косой композицией:

  $\forall L_1,L_2\subseteq Z^{\#}\ \forall\mathfrak{R}_1\ \forall\mathfrak{R}_2\ \forall\mathbf{S}_1\in\mathfrak{C}(L_{1\gamma})\ \forall\mathbf{S}_2\in\mathfrak{C}(L_{2\gamma})$

  $\cup\mathfrak{R}_1=L_1\ \&\ \cup\mathfrak{R}_2=L_2\ \Rightarrow\ \mathcal{T}_{\mathfrak{R}_1}(\mathbf{S}_1)\uparrow\downarrow\mathcal{T}_{\mathfrak{R}_2}(\mathbf{S}_2)\in\mathbf{S}_1\not\downarrow_{\mathfrak{R}_1,\mathfrak{R}_2}\not\downarrow\mathbf{S}_2$.

---

[25] Формально, сохранение эквивалентности означает: $a\sim_{\mathfrak{R}}b\ \Rightarrow\ \mathcal{T}_{\mathfrak{R}}(a)\sim_{\mathfrak{R}}b$. Это выполняется, поскольку, если $\mathcal{T}_{\mathfrak{R}}(a)\sim_{\mathfrak{R}}a$, то, по транзитивности эквивалентности, $\mathcal{T}_{\mathfrak{R}}(a)\sim_{\mathfrak{R}}a\ \&\ a\sim_{\mathfrak{R}}b\ \Rightarrow\ \mathcal{T}_{\mathfrak{R}}(a)\sim_{\mathfrak{R}}b$.

В этом случае семейство преобразований $\mathcal{T}$ будем называть монотонным (на классе $\mathfrak{C}$), а результат преобразования $\mathcal{T}_{\mathfrak{R}}(\mathbf{S})$ – монотонной спецификацией.

□

6.1.2. Восемь достаточных условий монотонности

В этом подразделе будем использовать понятие ϕ-трассы и ряд отношений и операций, не определяя их, а только указывая свойства, которыми они должны обладать.

Определение 80: Будем считать, что заданы:

- Отношение «один ко многим» $\preccurlyeq_{\mathfrak{R}} \subseteq \mathtt{L}_{\mathcal{P}(\mathtt{L})\Delta\gamma}{}^{*} \times \mathcal{P}(\mathtt{L}_{\mathcal{P}(\mathtt{L})\Delta\gamma}{}^{*})$, которое определяется для каждого алфавита $\mathtt{L} \subseteq \mathtt{Z}^{\#}$ и каждой $\mathfrak{R}$-семантики, где $\cup\mathfrak{R}=\mathtt{L}$, и называется $\mathfrak{R}$-мажорированием $\boldsymbol{F}$-трасс. Если $\mathtt{a} \preccurlyeq_{\mathfrak{R}} \mathtt{B}$, то будем говорить, что $\mathtt{B}$ $\mathfrak{R}$-мажорирует $\mathtt{a}$ и $\mathtt{a}$ $\mathfrak{R}$-мажорируется $\mathtt{B}$.
- Множество объектов $\boldsymbol{\Phi}^{\omega}(\mathtt{L})$, которое определяется для каждого алфавита $\mathtt{L} \subseteq \mathtt{Z}^{\#}$, элементы которого мы будем называть ϕ-трассами в этом алфавите. Множества ϕ-трасс в разных алфавитах могут пересекаться, то есть одна и та же ϕ-трасса может рассматриваться в нескольких алфавитах. Предполагается, что ϕ-трассы отличны от LTS-моделей $\boldsymbol{LTS}(\mathtt{L}_{\gamma}) \cap \boldsymbol{\Phi}^{\omega}(\mathtt{L}) = \varnothing$.[26]
- Отображение $\boldsymbol{\Phi} : \boldsymbol{LTS}(\mathtt{L}_{\gamma}) \to \mathcal{P}\circ\boldsymbol{\Phi}^{\omega}(\mathtt{L})$, которое для каждой LTS-модели в алфавите $\mathtt{L} \subseteq \mathtt{Z}^{\#}$ определяет множество её ϕ-трасс в этом алфавите.

---

[26] Мы требуем различения ϕ-трасс и LTS-моделей по техническим причинам: мы перегружаем оператор параллельной композиции, используя одно и то же обозначение $\between$ как для ϕ-трас, так и для LTS-моделей.

- Оператор $\xi : \mathbf{\Phi}^{\omega}(\mathrm{L}) \to \mathrm{L}_{\mathcal{P}(\mathrm{L})\Delta\gamma}{}^{*}$, который каждой ϕ-трассе в алфавите $\mathrm{L} \subseteq \mathrm{Z}^{\#}$ ставит в соответствие множество ***F***-трасс в этом алфавите.
- Отображение $. \downharpoonright\!\downharpoonright . : \mathbf{\Phi}^{\omega}(\mathrm{L}_1) \times \mathbf{\Phi}^{\omega}(\mathrm{L}_2) \to \mathcal{P}\circ\mathbf{\Phi}^{\omega}(\mathrm{L}_1 \downharpoonright\!\downharpoonright \mathrm{L}_2)$, которое каждой паре ϕ-трасс в алфавитах $\mathrm{L}_1, \mathrm{L}_2 \subseteq \mathrm{Z}^{\#}$ ставит в соответствие множество ϕ-трасс в композиции этих алфавитов и называется композицией ϕ-трасс.
- Отношение «один ко многим» $\preccurlyeq \subseteq \mathbf{\Phi}^{\omega}(\mathrm{L}) \times \mathcal{P}\circ\mathbf{\Phi}^{\omega}(\mathrm{L})$, которое определяется для каждого алфавита $\mathrm{L} \subseteq \mathrm{Z}^{\#}$ и называется мажорированием ϕ-трасс. Если $\mathrm{a} \preccurlyeq \mathrm{B}$, то будем говорить, что B мажорирует a и a мажорируется B.
- Отношения мажорирования распространим на случай «многие ко многим»: левое множество мажорируется правым множеством, если каждый элемент левого множества мажорируется правым множеством:

  $\mathrm{A} \preccurlyeq_{\mathfrak{R}} \mathrm{B} =_{\text{def}} \forall \mathrm{a} \in \mathrm{A}\ \mathrm{a} \preccurlyeq_{\mathfrak{R}} \mathrm{B}$,

  $\mathrm{A} \preccurlyeq \mathrm{B} =_{\text{def}} \forall \mathrm{a} \in \mathrm{A}\ \mathrm{a} \preccurlyeq \mathrm{B}$.

□

Определение 81: Будем говорить, что спецификация *мажорантна* (для подразумеваемой $\mathfrak{R}$-семантики), если множество её ϕ-трасс мажорирует множество ϕ-трасс каждой реализации, конформной этой спецификации по отношению $\boldsymbol{saco}_{\mathfrak{R}}$. Семейство преобразований $\mathcal{T}$ будем называть *мажорантным* (на классе $\mathfrak{C}$), если для каждой $\mathfrak{R}$-семантики преобразование $\mathcal{T}_{\mathfrak{R}}$ преобразует спецификацию (из класса $\mathfrak{C}$) в мажорантную спецификацию. Семейство преобразований $\mathcal{T}$ будем называть *конформным* (на классе $\mathfrak{C}$), если для каждой $\mathfrak{R}$-семантики преобразование $\mathcal{T}_{\mathfrak{R}}$ преобразует спецификацию (из класса $\mathfrak{C}$) в спецификацию, конформную исходной спецификации.

□

Определение 82: Условия монотонности.

$\forall \mathtt{L}\subseteq \mathtt{Z}^{\#}$ $\forall \mathfrak{R}$ такого, что $\cup\mathfrak{R}=\mathtt{L}$:

Монотонность 1: $\mathfrak{R}$-мажорирование влечёт конформность:

$\forall \mathbf{I},\mathbf{S}\in \boldsymbol{LTS}(\mathtt{L}_\gamma)\ \ \boldsymbol{F}(\mathbf{I})\preccurlyeq_{\mathfrak{R}}\boldsymbol{F}(\mathbf{S})\ \Rightarrow\ \mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}}\ \mathbf{S}$.

Монотонность 2: Генеративность ϕ-трасс:

$\forall \mathbf{S}\in \boldsymbol{LTS}_\gamma\ \cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}(\mathbf{S})\ =\ \boldsymbol{F}(\mathbf{S})$.

Монотонность 3: Аддитивность ϕ-трасс относительно композиции:

$\forall \mathbf{S}_1,\mathbf{S}_2\in \boldsymbol{LTS}_\gamma\ \ \boldsymbol{\Phi}^{\omega}(\mathbf{S}_1\between\mathbf{S}_2)=\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1)\between\boldsymbol{\Phi}^{\omega}(\mathbf{S}_2))$.

Монотонность 4: Генеративность ϕ-мажорирования:

$\forall \mathbf{I},\mathbf{S}\in \boldsymbol{LTS}(\mathtt{L}_\gamma)\ \ \boldsymbol{\Phi}^{\omega}(\mathbf{I})\preccurlyeq\boldsymbol{\Phi}^{\omega}(\mathbf{S})\ \Rightarrow\ \cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}(\mathbf{I})\preccurlyeq_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}(\mathbf{S})$.

Монотонность 5: Композиционность ϕ-мажорирования на классе $\mathfrak{C}$:

$\forall \mathbf{I}_1,\mathbf{I}_2\in \boldsymbol{LTS}_\gamma\ \ \forall \mathbf{S}_1,\mathbf{S}_2\in\mathfrak{C}$

$\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1)\preccurlyeq\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1)\ \ \&\ \ \boldsymbol{\Phi}^{\omega}(\mathbf{I}_2)\preccurlyeq\boldsymbol{\Phi}^{\omega}(\mathbf{S}_2)$

$\Rightarrow\ \ \cup(\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1)\between\boldsymbol{\Phi}^{\omega}(\mathbf{I}_2))\preccurlyeq\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1)\between\boldsymbol{\Phi}^{\omega}(\mathbf{S}_2))$.

Монотонность 6: Конформность преобразований на классе $\mathfrak{C}$:

$\forall \mathbf{S}\in\mathfrak{C}(\mathtt{L})\ \ \mathcal{T}_{\mathfrak{R}}(\mathbf{S})\ \ \boldsymbol{saco}_{\mathfrak{R}}\ \mathbf{S}$.

Монотонность 7: Мажорантность преобразований на классе $\mathfrak{C}$:

$\forall \mathbf{S}\in\mathfrak{C}(\mathtt{L})\ \ \forall \mathbf{I}\in\mathfrak{I}_{\mathfrak{R}}(\mathbf{S})\ \ \boldsymbol{\Phi}^{\omega}(\mathbf{I})\preccurlyeq\boldsymbol{\Phi}^{\omega}\circ\mathcal{T}_{\mathfrak{R}}(\mathbf{S})$.

Монотонность 8: Рефлексивность ϕ-мажорирования на классе $\mathfrak{C}$:

$\forall \mathbf{S}\in\mathfrak{C}\ \ \boldsymbol{\Phi}^{\omega}(\mathbf{S})\preccurlyeq\boldsymbol{\Phi}^{\omega}(\mathbf{S})$.

□

Теорема 23: Для того, чтобы семейство преобразований $\mathcal{T}$ было монотонным на классе $\mathfrak{C}$ достаточно выполнения условий монотонности 1÷7.

Для того, чтобы семейство преобразований $\mathcal{T}$ было левомонотонным на классе $\mathfrak{C}$ достаточно выполнения условий монотонности 1÷8.

□495

Замечание: Если класс $\mathfrak{C}$ замкнут по композиции моделей $\wr$, то результат композиции может использоваться как операнд в дальнейших композициях с сохранением монотонности. 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 взаимодействия.

Если (лево-)монотонное преобразование существует, то, в силу его инвариантности, результат преобразования эквивалентен по конформности исходной спецификации. А тогда, по Теорема 18:, преобразование сохраняет как класс конформных, так и класс безопасных реализаций. Заметим, что эта ситуация в корне отличается от той, что рассматривалась в главе о пополнении спецификаций: при наличии Ω-кнопок эквивалентные по конформности спецификации могли иметь различные классы безопасных реализаций.

6.2. $\mathfrak{R}$-мажорирование и конформность

Структура раздела:

1. $\mathfrak{R}$-мажорирование
2. Эквивалентность конформности $\mathfrak{R}$-мажорированию

6.2.1. $\mathfrak{R}$-мажорирование

Для общей теории монотонности нам нужно $\mathfrak{R}$-мажорирование ***F***-трасс как отношение «один ко многим». Мы определим мажорирование $\mathfrak{R}$-трасс как отношение «один ко многим» (Рис.63), а затем определим $\mathfrak{R}$-мажорирование ***F***-трасс как мажорирование их $\mathfrak{R}$-проекций.

Определение 83: Мажорирование $\mathfrak{R}$-трасс. Пусть заданы $\mathfrak{R}$-трасса μ и множество $\mathfrak{R}$-трасс $\mathbf{\Sigma}$.

$\mu \preccurlyeq_{\mathfrak{R}} \mathbf{\Sigma} =_{\text{def}} \mu \in \mathbf{\Sigma} \vee \langle\gamma\rangle \in \mathbf{\Sigma} \vee \mu \preccurlyeq_{\mathfrak{R}}^{\Delta} \mathbf{\Sigma} \vee \mu \preccurlyeq_{\mathfrak{R}}^{\Delta\gamma} \mathbf{\Sigma} \vee \mu \preccurlyeq_{\mathfrak{R}}^{P} \mathbf{\Sigma} \vee \mu \preccurlyeq_{\mathfrak{R}}^{z} \mathbf{\Sigma}$.

$\mu \preccurlyeq_{\mathfrak{R}}^{\Delta} \mathbf{\Sigma} =_{\text{def}} \exists \pi \ \pi \le \mu \ \& \ \mu \ne \pi \cdot \langle\gamma\rangle \ \& \ \pi \cdot \langle\Delta\rangle \in \mathbf{\Sigma}$.

$\mu \preccurlyeq_{\mathfrak{R}}^{\Delta\gamma} \mathbf{\Sigma} =_{\text{def}} \exists \pi \ \mu = \pi \cdot \langle\Delta\rangle \ \& \ \forall P \in \mathfrak{R} \ \exists t \in P \ \pi \cdot \langle t, \gamma\rangle \in \mathbf{\Sigma}$.

$\mu \preccurlyeq_{\mathfrak{R}}^{P} \mathbf{\Sigma} =_{\text{def}} \exists \pi \ \exists P \in \mathfrak{R} \ \pi \cdot \langle P\rangle \le \mu \ \& \ \exists z \in P \ \pi \cdot \langle z, \gamma\rangle \in \mathbf{\Sigma}$.

$\mu \preccurlyeq_{\mathfrak{R}}^{z} \mathbf{\Sigma} =_{\text{def}} \exists \pi \ \exists z \in L \ \pi \cdot \langle z\rangle \le \mu \ \& \ \forall P \in \mathfrak{R} \ (z \in P \Rightarrow \exists t \in P \ \pi \cdot \langle t, \gamma\rangle \in \mathbf{\Sigma})$.

□

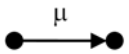
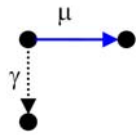
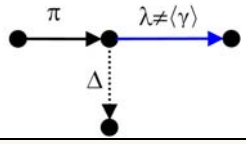
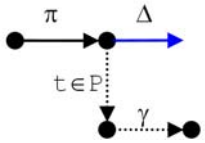
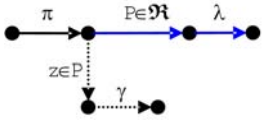
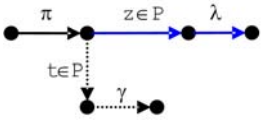
Мажорирование $\mathfrak{R}$-трасс $\mu \preccurlyeq_{\mathfrak{R}} \Sigma$		
$\mu \in \Sigma$	$\langle\gamma\rangle \in \Sigma$	$\mu=\pi\cdot\lambda \preccurlyeq_{\mathfrak{R}}^{\Delta} \Sigma$
		
$\mu=\pi\cdot\langle\Delta\rangle \preccurlyeq_{\mathfrak{R}}^{\Delta\gamma} \Sigma$	$\mu=\pi\cdot\langle P\rangle\cdot\lambda \preccurlyeq_{\mathfrak{R}}^{P} \Sigma$	$\mu=\pi\cdot\langle z\rangle\cdot\lambda \preccurlyeq_{\mathfrak{R}}^{z} \Sigma$
 $\forall P\in\mathfrak{R}\ \exists t$ …	 $\exists z$ …	 $\forall P\in\mathfrak{R}\ (z\in P \Rightarrow \exists t$ …)
⟶ префикс $\mathfrak{R}$-трассы μ, принадлежащий множеству Σ		
⟶ постфикс $\mathfrak{R}$-трассы μ (может не принадлежать множеству Σ)		
⋯⟶ продолжение префикса $\mathfrak{R}$-трассы μ, принадлежащее множеству Σ		

Рис.63.

Определение 84: $\mathfrak{R}$-мажорирование ***F***-трасс. Пусть заданы ***F***-трасса μ и множество ***F***-трасс Σ.

$\mu \preccurlyeq_{\mathfrak{R}} \Sigma =_{def} \mu^{\downarrow L_{\mathfrak{R}\Delta\gamma}} \preccurlyeq_{\mathfrak{R}} \Sigma^{\downarrow L_{\mathfrak{R}\Delta\gamma}}$.

□

6.2.2. Эквивалентность конформности $\mathfrak{R}$-мажорированию

Условие Монотонность 1: требует, чтобы $\mathfrak{R}$-мажорирование ***F***-трасс влекло конформность по отношению $saco_{\mathfrak{R}}$. Мы докажем их эквивалентность.

Теорема 24: LTS-реализация конформна LTS-спецификации тогда и только тогда, когда каждая ***F***-трасса реализации $\mathfrak{R}$-мажорируется ***F***-трассами спецификации:

$\forall \mathbf{I},\mathbf{S}\in \boldsymbol{LTS}(L_\gamma)\ \ \boldsymbol{F}(\mathbf{I}) \preccurlyeq_{\mathfrak{R}} \boldsymbol{F}(\mathbf{S}) \Leftrightarrow \mathbf{I}\ saco_{\mathfrak{R}}\ \mathbf{S}$.

В частности, выполнено условие Монотонность 1:.

6.3. ϕ-трассы и ϕ-модель

Структура раздела:

1. Определение ϕ-символов и ϕ-трасс
2. ϕ-трассы и ϕ-маршруты LTS
3. Нормальные ϕ-трассы и каноническая LTS
4. ϕ-модель

5. Эквивалентность LTS- и ϕ-моделей
6. ξ-оператор и генеративность ϕ-трасс
7. Композиция ϕ-трасс и аддитивность

6.3.1. Определение ϕ-символов и ϕ-трасс

Определение 85:

- ϕ-*символом* будем называть пару из двух непересекающихся подмножеств 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 в данном алфавите L, и обозначать

  $\Phi(L) =_{def} \{(a,b) \mid a\subseteq L \ \& \ b\subseteq L \ \& \ a\cap b=\varnothing\}$.

  Множество всех ϕ-символов обозначим $\Phi=\cup\{\Phi(L) \mid L\subseteq Z^{\#}\}$.
- Объединение алфавита и множества всех ϕ-символов в этом алфавите обозначим

  $L_\phi =_{def} L\cup\Phi(L)$.

□

Определение 86:

- *ϕ-трассой* в алфавите L будем называть (конечную или бесконечную) последовательность в алфавите базовых и ϕ-символов $L_{\phi\Delta\gamma}$.
- Максимальные префикс и постфикс ϕ-трассы $\boldsymbol{\sigma}$, состоящие только из ϕ-символов, будем обозначать:

  $\boldsymbol{pref}(\boldsymbol{\sigma}) =_{def} \langle\ \boldsymbol{\sigma}(i)\ \mid\ \boldsymbol{\sigma}[1..i] \in \Phi(L)^*\ \rangle$ и

  $\boldsymbol{postf}(\boldsymbol{\sigma}) =_{def} \langle\ \boldsymbol{\sigma}(i)\ \mid\ \boldsymbol{\sigma}[i..|\boldsymbol{\sigma}|] \in \Phi(L)^*\ \rangle$.

□

Поскольку в диссертации внешние действия, разрушение и дивергенция считаются праэлементами, в ϕ-трассах они отличимы от ϕ-символов как элементов декартового произведения.

Мы будем обозначать ϕ-трассы строчными жирными греческими буквами $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma}$…, а множества ϕ-трасс прописными жирными греческими буквами $\boldsymbol{\Lambda}, \boldsymbol{M}, \boldsymbol{\Sigma}$, ….

Одна и та же ϕ-трасса может рассматриваться в разных алфавитах внешних действий. Там, где это нужно, мы будем явно указывать алфавит, в котором рассматривается ϕ-трасса.

6.3.2. ϕ-трассы и ϕ-маршруты LTS

Структура подраздела:

- ϕ-символ стабильного состояния
- ϕ-трассы LTS
- Распространение LTS-обозначений на ϕ-трассы
- ϕ-маршруты LTS

ϕ-символ стабильного состояния

ϕ-символ стабильного состояния s – это пара множеств: ***ref***-множество = множество внешних действий, по которым нет переходов из состояния s, то

есть $L\backslash \mathit{init}(s)$, и ***gamma***-множество = множество внешних действий, по которым из s есть переходы в постсостояния с γ-переходами: $s\text{—}z\rightarrow s`\text{—}\gamma\rightarrow$. Такие внешние действия можно назвать *непосредственно-разрушающими* в состоянии.

Определение 87: Пусть задана LTS-модель $\mathbf{S}$. Для стабильного состояния $s\in V_{\mathbf{S}}$ определим:

- ***ref***-множество как множество отвергаемых внешних действий

  $\phi(s)_r =_{def} L\backslash \mathit{init}(s)$,

- ***gamma***-множество как множество непосредственно разрушающих внешних действий

  $\phi(s)_g =_{def} \{z\in L \mid \exists s` \; s\text{—}z\rightarrow s`\text{—}\gamma\rightarrow\}$.

- 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 внешних действий в состоянии s:

1) Действие z непосредственно-разрушающее, если есть переход в постсостояние с γ-переходом: $s\text{—}z\rightarrow s`\text{—}\gamma\rightarrow$.
2) Действие z (просто) разрушающее, если есть переход в постсостояние с γ-трассой: $s\text{—}z\rightarrow s`=\langle\gamma\rangle\Rightarrow$. 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.
3) Действие z разрушающее в трассовом смысле, если в состоянии s начинается $\mathfrak{R}$-трасса $\langle z,\gamma\rangle$: $s=\langle z\rangle\Rightarrow s`=\langle\gamma\rangle\Rightarrow$. В отличие от 2-го вида

здесь z-переход может начинаться не в самом состоянии s, а в каком-то другом состоянии, достижимом из s по непустой цепочке τ-переходов.

Очевидно, $1 \Rightarrow 2 \Rightarrow 3$, но обратные импликации, вообще говоря, не верны.

Для ϕ-символа состояния мы выбираем трактовку 1) – непосредственно разрушающие действия, поскольку в этом случае ***gamma***-множество стабильного 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 ведут из композиционного постсостояния перехода по действию z в композиционные состояния с γ-переходами. Трактовка 3) совпадает с трактовкой 2) для стабильного состояния s, поскольку из такого состояния не ведут τ-переходы, а для нестабильного состояния мы не определяем ϕ-символ. Таким образом, для стабильного состояния s из $z \in \phi(s)_g$ следует наличие в этом состоянии трассы $\langle z, \gamma \rangle$, но обратное, вообще говоря, не верно.

ϕ-трассы LTS

Выше мы вводили множество ***F***-трасс LTS через преобразование $\boldsymbol{L2L_F}: \boldsymbol{LTS}(L_\gamma) \rightarrow \boldsymbol{LTS}(L_{\mathcal{P}(L)\Delta\gamma})$ и взятие множества трасс полученной LTS. Аналогично множество ϕ-трасс LTS определим через преобразование $\boldsymbol{L2L_\phi}: \boldsymbol{LTS}(L_\gamma) \rightarrow \boldsymbol{LTS}(L_{\phi\Delta\gamma})$ и взятие множества трасс полученной LTS. Так же, как преобразование $\boldsymbol{L2L_F}$ отказ преобразует в переход-петлю, преобразование $\boldsymbol{L2L_\phi}$ в каждом стабильном состоянии s добавляет переход-петлю по символу $\phi(s)$. Оба преобразования все γ-переходы перенаправляют в специальное терминальное состояние **t**, и в это же состояние проводятся Δ-переходы из дивергентных состояний исходной LTS. Заметим, что для обоих преобразований в трассе преобразованной LTS разрушение и дивергенция могут быть только конечными символами.

Определение 88: Для алфавита L определим преобразование $\boldsymbol{L2L_\phi}: \boldsymbol{LTS}(L_\gamma) \to \boldsymbol{LTS}(L_{\phi\Delta\gamma})$. Для $\mathbf{S}=LTS(V_{\mathbf{S}}, L_\gamma, E_{\mathbf{S}}, s_0)$ $\boldsymbol{L2L_\phi}(\mathbf{S})=\mathbf{M}=LTS(V_{\mathbf{S}} \cup \{\mathbf{t}\}, L_{\phi\Delta\gamma}, E_{\mathbf{M}}, s_0)$, где $\mathbf{t} \notin V_{\mathbf{S}}$,[27] а $E_{\mathbf{M}}$ – наименьшее множество, порождаемое следующими правилами вывода:

$\forall s, s` \in V_{\mathbf{S}}$ $\forall z$

$z \in L_\tau$ & $s \text{—}z\to s` \vdash s \text{—}z\to s`$;

$s \text{—}\tau\gamma\not\to \vdash s \text{—}\phi(s)\to s$;

$s \text{—}\gamma\to \vdash s \text{—}\gamma\to \mathbf{t}$.

$s\uparrow \vdash s \text{—}\Delta\to \mathbf{t}$.

□

Трассы LTS $\boldsymbol{L2L_\phi}(\mathbf{S})$ – это и есть ϕ-трассы LTS $\mathbf{S}$. В отличие от $\boldsymbol{F}$-трасс мы будем рассматривать не только конечные, но и бесконечные ϕ-трассы.

Определение 89: ϕ-трассой LTS $\mathbf{S}$ будем называть трассу LTS $\boldsymbol{L2L_\phi}(\mathbf{S})$. Для состояния $s \in V_{\mathbf{S}}$

- $\boldsymbol{\Phi}(s) =_{def} \boldsymbol{T}(s)$ в $\boldsymbol{L2L_\phi}$

  – конечные ϕ-трассы с началом в состоянии s,

- $\boldsymbol{\Phi}^\infty(s) =_{def} \boldsymbol{T}^\infty(s)$ в $\boldsymbol{L2L_\phi}$

  – бесконечные ϕ-трассы с началом в состоянии s,

- $\boldsymbol{\Phi}^\omega(s) =_{def} \boldsymbol{T}^\omega(s)$ в $\boldsymbol{L2L_\phi}$

  – все ϕ-трассы с началом в состоянии s,

- $\boldsymbol{\Phi}(\mathbf{S}) =_{def} \boldsymbol{\Phi}(s_0)$ – конечные ϕ-трассы LTS $\mathbf{S}$,

- $\boldsymbol{\Phi}^\infty(\mathbf{S}) =_{def} \boldsymbol{\Phi}^\infty(s_0)$ – бесконечные ϕ-трассы LTS $\mathbf{S}$,

- $\boldsymbol{\Phi}^\omega(\mathbf{S}) =_{def} \boldsymbol{\Phi}^\omega(s_0)$ – все ϕ-трассы LTS $\mathbf{S}$.

---

[27] Если LTS уже содержит состояние, которое называется $\mathbf{t}$, то перед преобразованием переименуем его, что даст изоморфную LTS.

· ϕ-трасса $\sigma \in \Phi^{\omega}(s)$ *разрушающая*, если некоторый её префикс продолжается разрушением: $\exists \mu \le \sigma \ \ \mu \cdot \langle \gamma \rangle \in \Phi(s)$.

□

Ещё раз отметим, что при построении ϕ-трасс мы различаем непосредственно разрушающие и просто разрушающие действия. Первые попадают в ***gamma***-множество ϕ-символа, а вторые нет. Тем самым две LTS, изображённые на Рис.64, имеют различные ϕ-трассы.

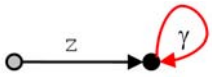
Две LTS с различными множествами ϕ-трасс	
	
Есть ϕ-трасса $\langle (\varnothing,\varnothing), z, \gamma \rangle$.	Нет ϕ-трассы $\langle (\varnothing,\varnothing), z, \gamma \rangle$.
Нет ϕ-трассы $\langle (\varnothing,\{z\}), z, \gamma \rangle$.	Есть ϕ-трасса $\langle (\varnothing,\{z\}), z, \gamma \rangle$.
$L=\{z\}$	

Рис.64.

Распространение LTS-обозначений на ϕ-трассы

Распространим LTS-обозначения (двойные стрелки и оператор ***after***) на ϕ-трассы аналогично тому, как это сделано для ***F***- и $\mathfrak{R}$-трасс.

Обозначение $s=\sigma\Rightarrow s'$ имеет один и тот же смысл для LTS $\mathbf{s}$ и $\boldsymbol{L2L}_{\phi}(\mathbf{s})$, если ϕ-трасса σ содержит только внешние действия. Если σ содержит ϕ-символы или заканчивается Δ-символом, двойная стрелка, очевидно, применяется к LTS $\boldsymbol{L2L}_{\phi}(\mathbf{s})$, поскольку в LTS $\mathbf{s}$ нет переходов по этим символам. Двусмысленность возможна лишь тогда, когда σ не содержит ϕ-символов и символа Δ и заканчивается разрушением γ, что происходит из-за того, что все γ-переходы в LTS $\boldsymbol{L2L}_{\phi}(\mathbf{s})$ перенаправляются в $\mathbf{t}$-состояние. Будем считать, что в этом случае имеется в виду LTS $\boldsymbol{L2L}_{\phi}(\mathbf{s})$. Тем самым, мы переопределяем двойную стрелку $s=\sigma\Rightarrow s'$ и производные обозначения для LTS $\mathbf{s}$ и трасс без отказов и Δ-символа, заканчивающихся разрушением.

Определение 90: Для $\mathbf{s}\in\boldsymbol{LTS}(L_\gamma)$, $s\in V_{\mathbf{s}}$, $s`\in V_{\mathbf{s}}\cup\{\mathbf{t}\}$, $\boldsymbol{\sigma}\in L_{\phi\Delta\gamma}{}^{\omega}$:

$s=\boldsymbol{\sigma}\Rightarrow s`$ $=_{def}$ $s=\boldsymbol{\sigma}\Rightarrow s`$ в LTS $\boldsymbol{L2L}_\phi(\mathbf{s})$ по Определению 35.

□

Одинарные стрелки $s\text{—}t\rightarrow s`$, $s\text{—}t\rightarrow$, $s\text{—}t\nrightarrow s`$ и $s\text{—}t\nrightarrow$, по-прежнему, означают наличие или отсутствие переходов и будут пониматься в контексте исходной LTS $\mathbf{s}$. В частности, $s=\langle\gamma\rangle\Rightarrow$ влечёт s ***after*** $\langle\gamma\rangle$ $=$ $\{\mathbf{t}\}$, но $s\text{—}\gamma\rightarrow s`$ влечёт $s`\neq\mathbf{t}$ (так как $\mathbf{t}\notin V_{\mathbf{s}}$).

ϕ-маршруты LTS

Определим ϕ-маршруты LTS аналогично $\mathfrak{R}$-маршрутам LTS.

Определение 91: ϕ-маршрутом в LTS $\mathbf{s}$ будем называть маршрут в LTS $\boldsymbol{L2L}_\phi(\mathbf{s})$. Для состояния $s\in V_{\mathbf{s}}$:

- $\boldsymbol{R}_\phi(s)$ $=_{def}$ $\boldsymbol{R}(s)$ в $\boldsymbol{L2L}_\phi(\mathbf{s})$

  – конечные ϕ-маршруты с началом в состоянии s,
- $\boldsymbol{R}_\phi{}^{\infty}(s)$ $=_{def}$ $\boldsymbol{R}^{\infty}(s)$ в $\boldsymbol{L2L}_\phi(\mathbf{s})$

  – бесконечные ϕ-маршруты с началом в состоянии s,
- $\boldsymbol{R}_\phi{}^{\omega}(s)$ $=_{def}$ $\boldsymbol{R}^{\omega}(s)$ в $\boldsymbol{L2L}_\phi(\mathbf{s})$

  – все ϕ-маршруты с началом в состоянии s,
- $\boldsymbol{R}_\phi(\mathbf{s})$ $=_{def}$ $\boldsymbol{R}_\phi(s_0)$ – конечные ϕ-маршруты LTS $\mathbf{s}$,
- $\boldsymbol{R}_\phi{}^{\infty}(\mathbf{s})$ $=_{def}$ $\boldsymbol{R}_\phi{}^{\infty}(s_0)$ – бесконечные ϕ-маршруты LTS $\mathbf{s}$,
- $\boldsymbol{R}_\phi{}^{\omega}(\mathbf{s})$ $=_{def}$ $\boldsymbol{R}_\phi{}^{\omega}(s_0)$ – все ϕ-маршруты LTS $\mathbf{s}$.

□

Определение 92:

- Для маршрута R LTS-модели $\mathbf{s}$ определим множество *ассоциированных* с ним ϕ-маршрутов, которые получаются из R следующей последовательностью операций:

1. первая обязательная операция: замена первого γ-перехода $s\text{—}\gamma\rightarrow s` \in \boldsymbol{Im}(R)$ на переход $s\text{—}\gamma\rightarrow \mathbf{t}$ и удаление всех последующих переходов маршрута R;
2. последовательность (быть может, пустая) вставок переходов вида $s\text{—}\phi(s)\rightarrow s$ в начало получившегося после предшествующих операций маршрута, если s начальное состояние маршрута, или после перехода с постсостоянием s, при условии (в обоих случаях), что s стабильное состояние;
3. последняя обязательная операция: замена бесконечного постфикса τ-переходов получившегося после предшествующих операций маршрута на моделирующий его Δ-переход в **t**-состояние.

· ϕ-трассой маршрута будем называть трассу ассоциированного с ним ϕ-маршрута. Множество таких конечных ϕ-трасс будем обозначать $\boldsymbol{\Phi}(R)$, бесконечных ϕ-трасс – $\boldsymbol{\Phi}^{\infty}(R)$ и всех ϕ-трасс – $\boldsymbol{\Phi}^{\omega}(R)$.

□

Сопоставление маршрута и ϕ-маршрутов отличается от аналогичной процедуры для ***F***-маршрутов только способом моделирования дивергенции Δ-переходом, что, в свою очередь, объясняется введением в рассмотрение бесконечных маршрутов и ϕ-маршрутов. Моделирующий дивергенцию Δ-переход ***F***-маршрута соответствует дивергентному конечному состоянию *конечного* маршрута S. Для ϕ-маршрута моделирующий дивергенцию Δ-переход соответствует бесконечному постфиксу τ-переходов *бесконечного* маршрута S.

Из определения множества ϕ-трасс маршрута вытекают очевидные следствия для множества ϕ-трасс состояния s (или LTS, то есть её начального состояния s_0):

$\boldsymbol{\Phi}(s) = \boldsymbol{T}\circ\boldsymbol{R}_{\phi}(s) = \cup\circ\boldsymbol{\Phi}\circ\boldsymbol{R}^{\omega}(s)$, $\boldsymbol{\Phi}(\mathbf{s}) = \boldsymbol{T}\circ\boldsymbol{R}_{\phi}(\mathbf{s}) = \cup\circ\boldsymbol{\Phi}\circ\boldsymbol{R}^{\omega}(\mathbf{s})$,

$\boldsymbol{\Phi}^{\infty}(s) = \boldsymbol{T}\circ\boldsymbol{R}_{\phi}^{\infty}(s) = \cup\circ\boldsymbol{\Phi}^{\infty}\circ\boldsymbol{R}^{\omega}(s)$, $\boldsymbol{\Phi}^{\infty}(\mathbf{s}) = \boldsymbol{T}\circ\boldsymbol{R}_{\phi}^{\infty}(\mathbf{s}) = \cup\circ\boldsymbol{\Phi}^{\infty}\circ\boldsymbol{R}^{\omega}(\mathbf{s})$,

$\boldsymbol{\Phi}^{\omega}(s) = \boldsymbol{T}\circ\boldsymbol{R}_{\phi}^{\omega}(s) = \cup\circ\boldsymbol{\Phi}^{\omega}\circ\boldsymbol{R}^{\omega}(s)$, $\boldsymbol{\Phi}^{\omega}(\mathbf{s}) = \boldsymbol{T}\circ\boldsymbol{R}_{\phi}^{\omega}(\mathbf{s}) = \cup\circ\boldsymbol{\Phi}^{\omega}\circ\boldsymbol{R}^{\omega}(\mathbf{s})$.

6.3.3. Нормальные ϕ-трассы и каноническая LTS

Структура подраздела:

- Нормальные ϕ-маршруты и ϕ-трассы
- Каноническая LTS

Нормальные ϕ-маршруты и ϕ-трассы

Определение 93:

- *Нормальным ϕ-маршрутом* LTS назовём такой ϕ-маршрут, который не содержит τ-циклов и непустого τ-постфикса (конечный ϕ-маршрут не заканчивается τ-переходом, а бесконечный ϕ-маршрут не имеет бесконечного постфикса τ-переходов) и при каждом проходе стабильного состояния его ϕ-петля проходится ровно один раз: каждый раз, когда при движении по ϕ-маршруту мы попадаем в стабильное состояние, мы проходим его ϕ-петлю один раз, после чего ϕ-маршрут заканчивается или продолжается переходом по базовому символу.
- Нормальной ϕ-трассой LTS будем называть такую ϕ-трассу, которая является трассой нормального ϕ-маршрута.
- Будем говорить, что LTS ϕ-детерминирована, если каждая её нормальная ϕ-трасса является трассой ровно одного нормального ϕ-маршрута.

□

Мы уже отмечали в начале этой главы, что ϕ-трассы похожи на трассы готовности, которые легко вычисляются по множеству ϕ-трасс LTS. В том же смысле нормальные ϕ-трассы похожи на нормальные трассы готовности, в которых при проходе состояния его множество готовности включается в трассу ровно один раз [89].

Нормальные ϕ-трассы удобны тем, что каждый маршрут имеет ровно одну нормальную ϕ-трассу, хотя может иметь много других ϕ-трасс. Это позволит нам в дальнейшем строить ϕ-детерминированные LTS.

Заметим, что, вообще говоря, нормальная ϕ-трасса не то же самое, что ϕ-трасса, в которую каждый ϕ-символ входит не более одного раза. В нормальную ϕ-трассу ϕ-символ стабильного состояния входит обязательно, а в ϕ-трассу без повторных ϕ-символов он может не входить. В частности, пустая ϕ-трасса будет нормальной только в том случае, когда начальное состояние LTS нестабильно. В то же время любую LTS $\mathbf{S}$ можно *нормализовать*, используя преобразование с сохранением множества ϕ-трасс в такую LTS $\mathbf{S}_{norm}$, в которой нормальные ϕ-трассы – это все её ϕ-трассы без повторных ϕ-символов. Для этого достаточно вместе с каждым её стабильным состоянием s добавить новое состояние s_{norm}, добавить переход $s \text{—}\tau\rightarrow s_{norm}$, и вместе с каждым переходом $s \text{—}z\rightarrow s'$ добавить переход $s_{norm} \text{—}z\rightarrow s'$.

Определение 94: *Нормальной* LTS будем называть LTS, в которой множество ϕ-трасс без повторных ϕ-символов совпадает с множеством нормальных ϕ-трасс. Определим преобразование LTS, которое будем называть нормализацией: для $\mathbf{S}=LTS(V_{\mathbf{S}}, L_\gamma, E_{\mathbf{S}}, s_0)$ выберем произвольный символ $_{norm}$ такой, что $\forall s\in V_{\mathbf{S}}$ $(s, _{norm})\notin V_{\mathbf{S}}$,[28] обозначим $s_{norm}=(s, _{norm})$, и определим $\mathbf{S}_{norm}=LTS(V_{\mathbf{S}}\times\{_{norm}\}, L_\gamma, E_{\mathbf{S}}\cup E_{norm}, s_0)$, где E_{norm} – наименьшее множество, порождаемое следующими правилами вывода: $\forall s, s', z$

$$s\in V_{\mathbf{S}} \;\&\; s \text{—}\tau\gamma\not\rightarrow \;\vdash\; s \text{—}\tau\rightarrow s_{norm},$$

$$s\in V_{\mathbf{S}} \;\&\; s \text{—}\tau\gamma\not\rightarrow \;\&\; z\in L \;\&\; s \text{—}z\rightarrow s' \;\vdash\; s_{norm} \text{—}z\rightarrow s'.$$

□

[28] Если $(s, _{norm})\in V_{\mathbf{S}}$, то перед преобразованием можно переименовать это состояние, что эквивалентно рассмотрению изоморфной LTS.

Лемма 34: 1) Множество конечных/всех ϕ-трасс LTS без повторных ϕ-символов получается из подмножества её нормальных конечных/всех ϕ-трасс с помощью операций удаления ϕ-символов. 2) Множество конечных/всех ϕ-трасс LTS получается из своего подмножества конечных/всех ϕ-трасс без повторных ϕ-символов с помощью операции повторения ϕ-символов.

□505

Лемма 35: 1) Преобразование нормализации сохраняет множество ϕ-трасс без повторных ϕ-символов. 2) Нормализованная LTS нормальна.

□505

Очевидно, что множество (конечных или всех) ϕ-трасс LTS однозначно определяет своё подмножество ϕ-трасс без повторных ϕ-символов. Однако соотношение нормальных ϕ-трасс и ϕ-трасс без повторных ϕ-символов не такое: две LTS могут иметь разные множества нормальных ϕ-трасс и одинаковые множества ϕ-трасс без повторных ϕ-символов. Преобразование нормализации, сохраняющее множество ϕ-трасс без повторных ϕ-символов, не сохраняет множество нормальных ϕ-трасс. Примером могут служить LTS $\mathbf{s}$ и $\mathbf{s}_{\mathrm{norm}}$, когда LTS $\mathbf{s}$ имеет стабильное начальное состояние, из которого выходит хотя бы один переход: в LTS $\mathbf{s}$ пустая ϕ-трасса не нормальна (начальное состояние стабильно), а в LTS $\mathbf{s}_{\mathrm{norm}}$ нормальна (начальное состояние нестабильно).

Каноническая LTS

Определение 95: LTS будем называть *канонической*, если выполняются следующие условия:

1) В каждом достижимом состоянии s определено не более одного перехода $s \text{—}u\rightarrow s`$ по каждому внешнему действию $u \in L$.

2) В каждом достижимом состоянии s определено не более одного τ-перехода $s \text{—}\tau\rightarrow s`$, ведущего в состояние $s`$, в котором определён γ-переход

$s` \text{—}\gamma\rightarrow$. При этом, если $s \text{—}\tau\rightarrow s` \text{—}\gamma\rightarrow$, то в состоянии $s`$ нет других переходов, а в состоянии s нет γ-перехода.

3) Все τ-переходы $s \text{—}\tau\rightarrow s_i$, определённые в достижимом нестабильном состоянии s и отличные от τ-петли $s \text{—}\tau\rightarrow s$ и τ-перехода $s \text{—}\tau\rightarrow s`$, ведущего в состояние с γ-переходом $s` \text{—}\gamma\rightarrow$, ведут в стабильные состояния, причём разные τ-переходы ведут в постсостояния с разными ϕ-символами: $s_i \neq s_j \Rightarrow \phi(s_i) \neq \phi(s_j)$.
4) Начальное состояние и постсостояния достижимых переходов по внешним действиям нестабильны.

□

Лемма 36: Каноническая LTS ϕ-детерминирована, а множество её нормальных ϕ-трасс префикс-замкнуто.

□507

Определение 96: Индуктивный предел. LTS будем называть *замкнутой по пределу* или *предельной*, если множество её ϕ-трасс предельно.

□

Лемма 37: Если каноническая LTS нормальна, то она предельна.

□511

6.3.4. ϕ-модель

Структура подраздела:

- Свойства ϕ-трасс и деревьев ϕ-трасс
- Определение ϕ-модели

Свойства ϕ-трасс и деревьев ϕ-трасс

Определение 97: Допустимость и согласованность.

- ϕ-трассу $\boldsymbol{\sigma}$ будем называть *допустимой*, если дивергенция и разрушение либо не входят в трассу, либо являются последним символом трассы: $\forall i\in[1..|\boldsymbol{\sigma}|-1]\ \boldsymbol{\sigma}(i)\notin\{\Delta,\gamma\}$.
- ϕ-трассу $\boldsymbol{\sigma}$ будем называть *согласованной*, если в ней за ϕ-символом следует либо тот же самый ϕ-символ, либо внешнее действие, не принадлежащее ***ref***-множеству этого ϕ-символа: $\forall i\in[1..|\boldsymbol{\sigma}|-1]$ $\boldsymbol{\sigma}(i)\in\phi(L) \Rightarrow \boldsymbol{\sigma}(i+1)=\boldsymbol{\sigma}(i) \vee \boldsymbol{\sigma}(i+1)\in L\setminus\boldsymbol{\sigma}(i)_r$.
- Для данного алфавита L множество конечных, бесконечных и всех допустимых и согласованных ϕ-трасс в этом алфавите обозначим, соответственно, $\boldsymbol{\Phi}(L)$, $\boldsymbol{\Phi}^{\infty}(L)$ и $\boldsymbol{\Phi}^{\omega}(L)$.
- Объединение множеств конечных, бесконечных и всех допустимых и согласованных ϕ-трасс во всех алфавитах обозначим, соответственно, $\boldsymbol{\Phi}=\cup\{\boldsymbol{\Phi}(L)\,|\,L\subseteq Z^{\#}\}$, $\boldsymbol{\Phi}^{\infty}=\cup\{\boldsymbol{\Phi}^{\infty}(L)\,|\,L\subseteq Z^{\#}\}$ и $\boldsymbol{\Phi}^{\omega}=\cup\{\boldsymbol{\Phi}^{\omega}(L)\,|\,L\subseteq Z^{\#}\}$.

□

Определение 98: Конвергентность. Пусть задано дерево ϕ-трасс $\boldsymbol{\Sigma}$. ϕ-трассу $\boldsymbol{\sigma}\in\boldsymbol{\Sigma}$ будем называть *конвергентной* (в дереве $\boldsymbol{\Sigma}$), и обозначать $\boldsymbol{\sigma}\downarrow$, если она продолжается в дереве $\boldsymbol{\Sigma}$ некоторым ϕ-символом. Если ϕ-трасса не конвергентна, будем называть её *дивергентной* (в дереве $\boldsymbol{\Sigma}$) и обозначать $\boldsymbol{\sigma}\uparrow$.

$\boldsymbol{\sigma}\downarrow =_{def} \exists o\in\Phi(L)\ \boldsymbol{\sigma}\cdot\langle o\rangle\in\boldsymbol{\Sigma}$; $\boldsymbol{\sigma}\uparrow =_{def} \neg\boldsymbol{\sigma}\downarrow$.

□

Определение 99: Удаление и повторение ϕ-символов.

· Определим следующие операции над ϕ-трассами:

  · ***d*** (***d***eletion) – удаление ϕ-символа:

    $\circ \in \Phi(L)$: $(\boldsymbol{\mu}\cdot\langle\circ\rangle\cdot\boldsymbol{\lambda}, \boldsymbol{\mu}) \text{—}\!\!*d*\!\!\rightarrow \boldsymbol{\mu}\cdot\boldsymbol{\lambda}$.

  · ***r*** (***r***epetition) – повторение ϕ-символа:

    $\circ \in \Phi(L)$: $(\boldsymbol{\mu}\cdot\langle\circ\rangle\cdot\boldsymbol{\lambda}, \boldsymbol{\mu}) \text{—}\!\!*r*\!\!\rightarrow \boldsymbol{\mu}\cdot\langle\circ,\circ\rangle\cdot\boldsymbol{\lambda}$.

· Определим замыкание по набору таких операций допустимой и согласованной ϕ-трассы $\boldsymbol{\sigma}$:

  · $\boldsymbol{d}(\boldsymbol{\sigma})$ – множество ϕ-трасс, отличающихся от $\boldsymbol{\sigma}$ только, быть может, меньшим числом ϕ-символов в начале и после каждого внешнего действия, где в $\boldsymbol{\sigma}$ имелся ϕ-символ;

  · $\boldsymbol{r}(\boldsymbol{\sigma})$ – множество ϕ-трасс, отличающихся от $\boldsymbol{\sigma}$ только, быть может, большим числом ϕ-символов в начале и после каждого внешнего действия, где в $\boldsymbol{\sigma}$ имелся ϕ-символ;

  · $\boldsymbol{dr}(\boldsymbol{\sigma})$ – множество ϕ-трасс, отличающихся от $\boldsymbol{\sigma}$ только, быть может, другим числом ϕ-символов в начале и после каждого внешнего действия, где в $\boldsymbol{\sigma}$ имелся ϕ-символ.

□

Определение 100: Продолжение ϕ-трассы после ϕ-символа.

· Определим операцию $\boldsymbol{a}$ продолжения ϕ-трассы, заканчивающейся на ϕ-символ $\circ$ внешним действием $z \notin \circ_r$ и далее разрушением, если $z \in \circ_g$:

  $\circ \in \Phi(L)$ и $z \in L \setminus \circ_r$: $\boldsymbol{\mu}\cdot\langle\circ\rangle \text{—}\!\!*a*\!\!\rightarrow \boldsymbol{\mu}\cdot\langle\circ, z\rangle$,

  $\circ \in \Phi(L)$ и $z \in \circ_g$: $\boldsymbol{\mu}\cdot\langle\circ\rangle \text{—}\!\!*a*\!\!\rightarrow \boldsymbol{\mu}\cdot\langle\circ, z, \gamma\rangle$,

· Определим замыкание по операции $\boldsymbol{a}$:

  $\boldsymbol{a}(\boldsymbol{\mu}\cdot\langle\circ\rangle) =_{def} \{\boldsymbol{\mu}\cdot\langle\circ, z\rangle \mid z \in L \setminus \circ_r\} \cup \{\boldsymbol{\mu}\cdot\langle\circ, z, \gamma\rangle \mid z \in \circ_g\}$.

□

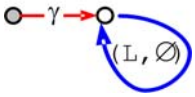
Определение ϕ-модели

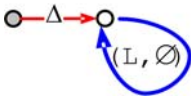
Определение 101: Определение ϕ-модели.

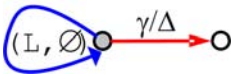
- Непустое дерево ϕ-трасс $\mathbf{\Sigma}$ будем называть *предельной ϕ-моделью*, если выполнены следующие шесть требований:
  - (ϕ1) допустимость: все ϕ-трассы $\mathbf{\Sigma}$ допустимы;
  - (ϕ2) согласованность: все ϕ-трассы $\mathbf{\Sigma}$ согласованы;
  - (ϕ3) конвергентность: все ϕ-трассы $\mathbf{\Sigma}$, не содержащие и не продолжающиеся разрушением и дивергенцией, конвергентны;
  - (ϕ4) замкнутость: $\mathbf{\Sigma}$ замкнуто по ***dr***-операциям: $\cup_{\circ}\boldsymbol{dr}(\mathbf{\Sigma})=\mathbf{\Sigma}$;
  - (ϕ5) полнота: $\mathbf{\Sigma}$ замкнуто по ***a***-операции: $\cup_{\circ}\boldsymbol{a}(\mathbf{\Sigma})=\mathbf{\Sigma}$.
  - (ϕ6) предельность: множество $\mathbf{\Sigma}$ предельно.
- Множество ϕ-трасс $\mathbf{\Sigma}$ будем называть *ϕ-моделью*, если оно представимо в виде объединения множества предельных ϕ-моделей.
- Множество всех ϕ-моделей в алфавите L обозначим $\boldsymbol{\Phi MODEL}(L)$.

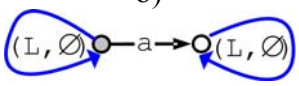
□

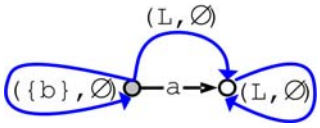
Шесть свойств ϕ-модели независимы в следующем смысле: существует алфавит внешних действий $L\subseteq Z^{\#}$ такой, что для некоторых деревьев ϕ-трасс могут выполняться все свойства, кроме одного. Примеры показаны на Рис.65. Здесь для каждого i-го свойства ($i=1\div5$) изображён порождающий граф дерева ϕ-трасс, удовлетворяющего всем свойствам ϕ-модели, кроме i-го. Для свойства предельности приведён пример дерева ϕ-трасс, в котором каждая ϕ-трасса является префиксом максимальной по длине ϕ-трассы, но дерево не замкнуто по пределу и не представимо в виде объединения предельных ϕ-моделей.

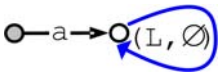
<table>
<tr><th colspan="3">Независимость свойств ϕ-модели</th></tr>
<tr><td>1. допустимость
L
a)
 

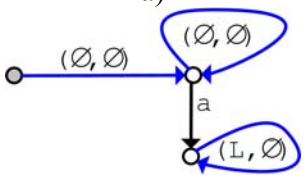
b)
 

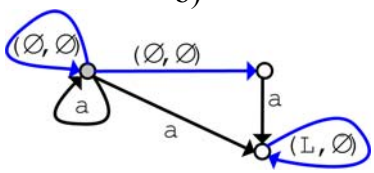
не допустимы ϕ-трассы
a)⟨γ,(L,∅)⟩
b)⟨Δ,(L,∅)⟩</td><td>2. согласованность
L={a,b}
a)
 

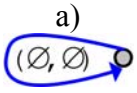
b)
 

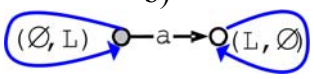
c)
 

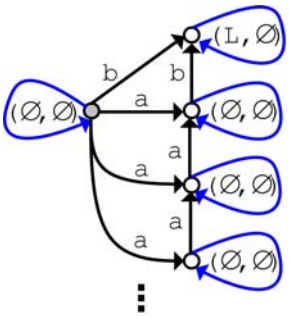
не согласованы ϕ-трассы
a)⟨(L,∅),γ⟩, ⟨(L,∅),Δ⟩
b)⟨(L,∅),a,(L,∅)⟩
c)⟨({b},∅),(L,∅)⟩</td><td>3. конвергентность
L={a}
 

ϵ↓</td></tr>
<tr><td>4. замкнутость</td><td>5. полнота</td><td>6. предельность</td></tr>
<tr><td>L={a}
a)
 

b)
 

a) нарушение d-замкнутости: нет ϕ-трассы ⟨a⟩
b) нарушение r-замкнутости: нет ϕ-трассы ⟨(∅,∅),(∅,∅),a,(L,∅)⟩</td><td>L={a}
a)
 

b)
 

нет ϕ-трасс
a)⟨(∅,∅),a⟩
b)⟨(∅,L),a,γ⟩</td><td>L={a,b}
 

отсутствует предел ⟨x,x,x,…⟩,
где x=(∅,∅),a</td></tr>
</table>

Рис.65.

6.3.5. Эквивалентность LTS- и ϕ-моделей

Структура подраздела:

· Множество ϕ-трасс LTS-модели как ϕ-модель

· ϕ-модель как множество ϕ-трасс LTS-модели

Мы покажем, что множество ϕ-трасс LTS-модели является ϕ-моделью и, наоборот, для каждой ϕ-модели существует LTS-модель, имеющая то же множество ϕ-трасс.

Множество ϕ-трасс LTS-модели как ϕ-модель

Для каждой LTS можно построить LTS-дерево, состояниями которого будут конечные маршруты исходной LTS.

Определение 102: Для LTS-модели $\mathbf{S}=\mathrm{LTS}(V_{\mathbf{S}}, L_{\gamma}, E_{\mathbf{S}}, s_0)$ определим *маршрутную* LTS $\boldsymbol{Run}(\mathbf{S})=\mathrm{LTS}(\boldsymbol{R}(\mathbf{S}), L_{\gamma}, E_{\boldsymbol{Run}(\mathbf{S})}, (s_0, \epsilon))$, где $E_{\boldsymbol{Run}(\mathbf{S})}$ – наименьшее множество, порождаемое следующим правилом вывода:

$\forall P \in \boldsymbol{R}(\mathbf{S})\ \ \forall u \in L_{\gamma\tau}\ \ \forall s \in V_{\mathbf{S}}$

$\omega(P) \text{—}u\rightarrow s \ \ \vdash\ P \text{—}u\rightarrow P\cdot\langle(\omega(P), u, s)\rangle$.

□

Лемма 38: Для каждой LTS-модели $\mathbf{S}$ её маршрутная LTS является деревом и имеет то же множеством ϕ-трасс, что LTS $\mathbf{S}$.

□512

Заменяя LTS-модель на её маршрутную LTS и выделяя в этой маршрутной LTS ЛК-ветвящиеся поддеревья с сохранением ϕ-символов состояний, получаем следующее утверждение.

Лемма 39: Для каждой LTS-модели $\mathbf{S}$ существует множество ЛК-ветвящихся LTS-моделей, объединение множеств ϕ-трасс которых равно множеству ϕ-трасс LTS $\mathbf{S}$.

□512

Лемма 40: Множество ϕ-трасс ЛК-ветвящейся LTS $\mathbf{s}$ предельно.

□513

Теорема 25: Множество ϕ-трасс LTS является ϕ-моделью.

□514

ϕ-модель как множество ϕ-трасс LTS-модели

Сначала по предельной ϕ-модели построим каноническую LTS-модель, имеющую то же множество ϕ-трасс (Рис.66). Её состояниями будут ϕ-трассы ϕ-модели без повторных ϕ-символов, не заканчивающиеся на дивергенцию и разрушение, а также специальное новое состояние $\mathbf{t}$. Начальное состояние – пустая ϕ-трасса. Продолжение ϕ-трассы внешним действием соответствует переходу по этому действию, а продолжение ϕ-трассы ϕ-символом – τ-переход в состояние, соответствующее продолженной ϕ-трассе. Дивергенция соответствует τ-петле.

Что касается разрушения, то мы не можем просто определить γ-петлю в состоянии, соответствующем трассе, продолжающейся разрушением. Дело в том, что ***gamma***-множество ϕ-символа состоит из непосредственно разрушающих действий, а не просто разрушающих. В частности, различаются множества ϕ-трасс двух LTS на Рис.64. Такую γ-петлю мы можем определять только в состоянии, соответствующем ϕ-трассе, заканчивающейся ϕ-символом и далее действием из его ***gamma***-множества. Если же ϕ-трасса не имеет такого вида, но продолжается разрушением, то мы должны определять τ-переход в некоторое другое состояние, в котором уже определять γ-петлю. Таким состоянием у нас будет новое состояние $\mathbf{t}$.

Определение 103: Определим преобразование ***Φ2L*** ϕ-модели $\mathbf{\Sigma}$ в LTS-модель следующим образом: $\boldsymbol{\Phi 2L}(\mathbf{\Sigma})=\mathbf{s}=\mathtt{LTS}(\mathtt{V}_{\mathbf{s}},\mathtt{L}_{\gamma},\mathtt{E}_{\mathbf{s}},\epsilon)$, где $\mathtt{V}_{\mathbf{s}}=\{\mathbf{t}\}\cup\{\boldsymbol{\mu}\in\mathbf{\Sigma}\,|\,\gamma,\Delta\notin \boldsymbol{Im}(\boldsymbol{\mu})$ & $\boldsymbol{\mu}$ не имеет повторных ϕ-символов$\}$, где $\mathbf{t}\notin\mathbf{\Sigma}$,

а $E_{\mathbf{s}}$ – наименьшее множество, порождаемое следующими правилами вывода:

$\forall \boldsymbol{\mu}\ \forall z \in L\ \forall o \in \Phi(L)$

(1) $\boldsymbol{\mu}\cdot\langle z\rangle \in V_{\mathbf{s}} \vdash \boldsymbol{\mu} \text{—} z \rightarrow \boldsymbol{\mu}\cdot\langle z\rangle$;

(2) $\boldsymbol{\mu}\cdot\langle o\rangle \in V_{\mathbf{s}} \vdash \boldsymbol{\mu} \text{—} \tau \rightarrow \boldsymbol{\mu}\cdot\langle o\rangle$;

(3) $\boldsymbol{\mu} \in V_{\mathbf{s}}$ & $\boldsymbol{\mu}\cdot\langle\Delta\rangle \in \boldsymbol{\Sigma} \vdash \boldsymbol{\mu} \text{—} \tau \rightarrow \boldsymbol{\mu}$;

(4) $\boldsymbol{\mu}\cdot\langle o\rangle \in V_{\mathbf{s}}$ & $z \in o_{\mathbf{g}} \vdash \boldsymbol{\mu}\cdot\langle o, z\rangle \text{—} \gamma \rightarrow \boldsymbol{\mu}\cdot\langle o, z\rangle$;

(5) $\forall \boldsymbol{\lambda}\ \neg(\boldsymbol{\mu} = \boldsymbol{\lambda}\cdot\langle o, z\rangle$ & $z \in o_{\mathbf{g}})$ & $\boldsymbol{\mu}\cdot\langle\gamma\rangle \in \boldsymbol{\Sigma} \vdash \boldsymbol{\mu} \text{—} \tau \rightarrow \mathbf{t} \text{—} \gamma \rightarrow \mathbf{t}$.

□

Преобразование ***Φ2L***

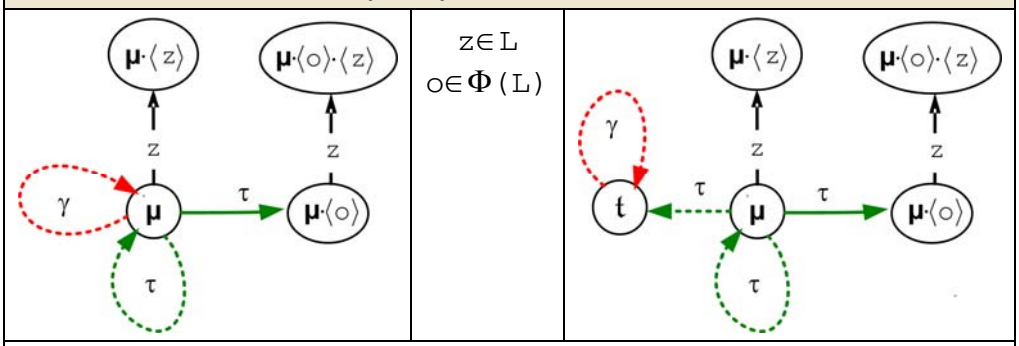


Рис.66.

Лемма 41: Для ϕ-модели $\boldsymbol{\Sigma}$ в LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ 1) каждое состояние $\boldsymbol{\mu}$, где $\boldsymbol{postf}(\boldsymbol{\mu}) = \epsilon$, нестабильно; 2) каждое состояние $\boldsymbol{\mu}\cdot\langle o\rangle$, где $o \in \Phi(L)$, стабильно и $\phi(\boldsymbol{\mu}\cdot\langle o\rangle) = o$.

□515

Лемма 42: Для ϕ-модели $\boldsymbol{\Sigma}$ LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ каноническая.

□516

Лемма 43: Для ϕ-модели $\boldsymbol{\Sigma}$ в LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ 1) в каждом состоянии $\boldsymbol{\mu}$ заканчивается нормальная ϕ-трасса $\boldsymbol{\mu}$; 2) каждый конечный нормальный ϕ-маршрут с трассой $\boldsymbol{\mu}$, не завершающейся символом Δ или γ, заканчивается в единственном состоянии $\boldsymbol{\mu}$; 3) имеется конечная нормальная ϕ-трасса $\boldsymbol{\mu}\cdot\langle\Delta\rangle$

или $\boldsymbol{\mu}\cdot\langle\gamma\rangle$ тогда и только тогда, когда в $\boldsymbol{\Sigma}$ имеется ф-трасса без повторных ф-символов $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ или $\boldsymbol{\mu}\cdot\langle\gamma\rangle$, соответственно.

□517

Лемма 44: Множество конечных ф-трасс ф-модели $\boldsymbol{\Sigma}$ совпадает с множеством конечных ф-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$.

□520

Лемма 45: Для ф-модели $\boldsymbol{\Sigma}$ LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ нормальная.

□520

Лемма 46: Для ф-модели $\boldsymbol{\Sigma}$ LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ предельная.

□521

Лемма 47: Предельная ф-модель $\boldsymbol{\Sigma}$ равна множеству ф-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$.

□521

Лемма 48: Объединение множества LTS в одном алфавите имеет множество ф-трасс равное объединению множеств ф-трасс LTS-компонентов.

□522

Теорема 26: ф-модель является множеством ф-трасс некоторой LTS.

□522

Мы показали, что каждой LTS-модели однозначно соответствует ф-модель. В то же время для ф-модели, очевидно, можно построить много LTS-моделей, имеющих то же множество ф-трасс.

6.3.6. ξ-оператор и генеративность ф-трасс

ξ-оператор определяет порождаемую (генерируемую) ***F***-трассу заменой каждого ф-символа ф-трассы на конечную последовательность отказов, вложенных в ***ref***-множество этого ф-символа. Подтрассы базовых символов – внешних действий, дивергенции и разрушения – одинаковы в ф-трассе и порождаемых ею ***F***-трассах. Заметим, что для генерации ***F***-трасс используются

только ***ref***-множества ϕ-символов ϕ-трассы, а их ***gamma***-множества не используются.

При таком определении порождаемых ***F***-трасс для получения всех ***F***-трасс LTS нам достаточно только конечных ϕ-трасс. Мы формально распространим ξ-оператор на бесконечные ϕ-трассы как возвращающий пустое множество ***F***-трасс.

Определение 104: Определим ξ-оператор, строящий по ϕ-трассе множество *порождаемых* ею ***F***-трасс.

- Сначала определим ξ-оператор для конечных ϕ-трасс.

  Для $o \in \Phi(L)$, $z \in L_{\Delta\gamma}$ и $\boldsymbol{\sigma} \in \boldsymbol{\Phi}(L)$

  - $\xi(o) =_{def} \mathcal{P}(o_r)^*$,
  - $\xi(z) =_{def} \{\langle z \rangle\}$,
  - $\xi(\boldsymbol{\sigma}) =_{def} \cdot(\xi \circ \boldsymbol{\sigma}) = \xi(\boldsymbol{\sigma}(1)) \cdot \ldots \cdot \xi(\boldsymbol{\sigma}(|\boldsymbol{\sigma}|))$.[29]

- Распространим ξ-оператор на бесконечные ϕ-трассы.

  Для $L \subseteq Z^{\#}$ и $\boldsymbol{\sigma} \in \boldsymbol{\Phi}^{\infty}(L)$

  - $\xi(\boldsymbol{\sigma}) =_{def} \varnothing$.

□

Теорема 27: Условие Монотонность 2: выполнено: ϕ-трассы генеративны: $\forall \mathbf{s} \in \boldsymbol{LTS}_{\gamma} \cup \circ \xi \circ \boldsymbol{\Phi}^{\omega}(\mathbf{s}) = \boldsymbol{F}(\mathbf{s})$.

□522

Мы показали, что каждой ϕ-модели однозначно соответствует $\mathfrak{R}$-модель. В то же время одной $\mathfrak{R}$-модели может соответствовать много ϕ-моделей, генерирующих то же множество $\mathfrak{R}$-трасс. С учётом соотношения ϕ-моделей и

---

[29] Конкатенация последовательности множеств последовательностей – это множество всех возможных конкатенаций последовательностей из этих множеств.

LTS-моделей имеем общую картину соотношения моделей, условно изображённую на Рис.67.

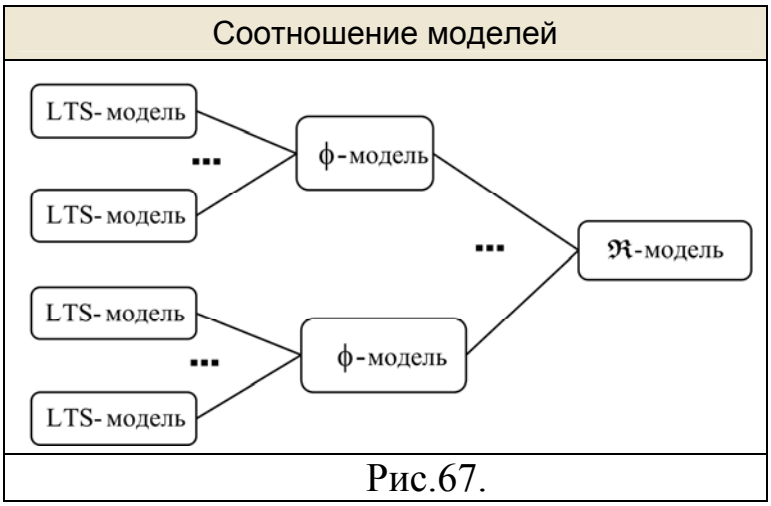


Рис.67.

Генеративность ϕ-трасс позволяет определить конформность на классе ϕ-моделей.

Определение 105: Будем говорить, что ϕ-реализация безопасна для (конформна) ϕ-спецификации, если отношением безопасности (конформностью) связаны генерируемые ими ***F***-модели: $\forall \mathbf{I}, \mathbf{\Sigma} \in \boldsymbol{\Phi MODEL}(\mathtt{L})$

- $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma} =_{\text{def}} \cup_{\circ}\xi(\mathbf{I})$ ***safe for*** $\cup_{\circ}\xi(\mathbf{\Sigma})$,
- $\boldsymbol{safe\mathfrak{I}}(\mathbf{\Sigma}) =_{\text{def}} \{\mathbf{I} \in \boldsymbol{\Phi MODEL}(\mathtt{L}) \mid \mathbf{I}$ ***safe for*** $\mathbf{\Sigma}\}$,
- $\mathbf{I}$ ***saco*** $\mathbf{\Sigma} =_{\text{def}} \cup_{\circ}\xi(\mathbf{I})$ ***saco*** $\cup_{\circ}\xi(\mathbf{\Sigma})$,
- $\mathfrak{I}(\mathbf{\Sigma}) =_{\text{def}} \{\mathbf{I} \in \boldsymbol{\Phi MODEL}(\mathtt{L}) \mid \mathbf{I}$ ***saco*** $\mathbf{\Sigma}\}$.

□

Там, где по контексту неясно, в какой $\mathfrak{R}$-семантике рассматриваются отношения и множества, мы будем указывать семантику в нижнем индексе и писать, соответственно: $\boldsymbol{safe}_{\mathfrak{R}}$ ***for***, $\boldsymbol{safe\mathfrak{I}}_{\mathfrak{R}}$, $\boldsymbol{saco}_{\mathfrak{R}}$ и $\mathfrak{I}_{\mathfrak{R}}$.

6.3.7. Композиция ϕ-трасс и аддитивность

Структура подраздела:

- Композиция ϕ-символов
- Композиция ϕ-трасс
- Композиция маршрутов
- Аддитивность ϕ-трасс относительно композиции

Композиция ϕ-символов

Определим композицию ϕ-символов так, чтобы ϕ-символ стабильного композиционного состояния был равен композиции ϕ-символов стабильных состояний-операндов. Если композиционное состояние нестабильно, оно не имеет ϕ-символа, что в композиции ϕ-символов будем изображать символом τ. Если состояния-операнды стабильны, композиционное состояние нестабильно только тогда, когда есть синхронный переход: в одном состоянии-операнде есть переход по внешнему действию z, а в другом – по противоположному действию $\underline{z}$. В терминах ϕ-символов это означает, что существуют синхронные внешние действия 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.

Определение 106: Пусть $A,B\subseteq Z^{\#}$. Определим композицию ϕ-символов $.\!\upharpoonright\!\downharpoonright.:\Phi(A)\times\Phi(B)\to\Phi(A\upharpoonright\!\downharpoonright B)_\tau$ следующим образом:

$\forall a\in\Phi(A)\ \ \forall b\in\Phi(B)$

$(A\setminus a_r)\cap(\underline{B\setminus b_r})\neq\varnothing$: $a\upharpoonright\!\downharpoonright b =_{def} \tau$,

иначе: $a\upharpoonright\!\downharpoonright b =_{def} ((A\setminus\underline{B})\cap a_r\cup(B\setminus\underline{A})\cap b_r, (A\setminus\underline{B})\cap a_g\cup(B\setminus\underline{A})\cap b_g)$

□

Лемма 49: Пусть $A,B\subseteq Z^{\#}$, $\boldsymbol{\kappa}\in \boldsymbol{LTS}(A_\gamma)$, $\boldsymbol{\iota}\in \boldsymbol{LTS}(B_\gamma)$, и пусть состояния $k\in V_{\boldsymbol{\kappa}}$ и $l\in V_{\boldsymbol{\iota}}$ стабильны. Тогда для состояния $kl\in V_{\boldsymbol{\kappa}\|\boldsymbol{\iota}}$ имеет место:

a) $kl\text{—}\tau\gamma\twoheadrightarrow \Leftrightarrow \phi(k)\uparrow\downarrow\phi(l)\neq\tau$,

b) $kl\text{—}\tau\gamma\twoheadrightarrow \Rightarrow \phi(kl)=\phi(k)\uparrow\downarrow\phi(l)$.

□523

Композиция ϕ-трасс

Определение 107: Пусть $A,B\subseteq Z^{\#}$. Определим композицию ϕ-трасс $.\uparrow\downarrow.:\boldsymbol{\Phi}^{\omega}(A)\times\boldsymbol{\Phi}^{\omega}(B)\to\boldsymbol{\Phi}^{\omega}(A\uparrow\downarrow B)$ как наименьшее множество последовательностей, порождаемое следующими правилами вывода:

$\forall\boldsymbol{\kappa}\in\boldsymbol{\Phi}(A)\quad \forall\boldsymbol{\lambda}\in\boldsymbol{\Phi}(B)\quad \forall\boldsymbol{\mu}\in\boldsymbol{\Phi}(A\uparrow\downarrow B)$

$\forall\boldsymbol{\kappa}`\in\boldsymbol{\Phi}^{\omega}(A)\quad \forall\boldsymbol{\lambda}`\in\boldsymbol{\Phi}^{\omega}(B)\quad \forall\boldsymbol{\mu}`\in\boldsymbol{\Phi}^{\omega}(A\uparrow\downarrow B)$

$(\uparrow_\phi\downarrow 0)$ начальное правило: $\vdash \epsilon \in \epsilon\uparrow\downarrow\epsilon$,

$(\uparrow_\phi\downarrow 1)$ асинхронная композиция с левым базовым символом:

$\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}\ \&\ z\in A_{\Delta\gamma}\setminus\underline{B}\ \&\ \Delta,\gamma\notin\boldsymbol{Im}(\boldsymbol{\mu})\ \vdash \boldsymbol{\mu}\cdot\langle z\rangle\in(\boldsymbol{\kappa}\cdot\langle z\rangle)\uparrow\downarrow\boldsymbol{\lambda}$,

$(\uparrow_\phi\downarrow 2)$ асинхронная композиция с правым базовым символом:

$\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}\ \&\ z\in B_{\Delta\gamma}\setminus\underline{A}\ \&\ \Delta,\gamma\notin\boldsymbol{Im}(\boldsymbol{\mu})\ \vdash \boldsymbol{\mu}\cdot\langle z\rangle\in\boldsymbol{\kappa}\uparrow\downarrow(\boldsymbol{\lambda}\cdot\langle z\rangle)$,

$(\uparrow_\phi\downarrow 3)$ синхронная композиция базовых символов:

$\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}\ \&\ z\in A\cap\underline{B}\ \vdash \boldsymbol{\mu}\in(\boldsymbol{\kappa}\cdot\langle z\rangle)\uparrow\downarrow(\boldsymbol{\lambda}\cdot\langle\underline{z}\rangle)$,

$(\uparrow_\phi\downarrow 4)$ синхронная композиция ϕ-символов:

$\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}\ \&\ a\in\Phi(A)\ \&\ b\in\Phi(B)\ \vdash \boldsymbol{\mu}\cdot\langle a\uparrow\downarrow b\rangle\uparrow\{\tau\}\in(\boldsymbol{\kappa}\cdot\langle a\rangle)\uparrow\downarrow(\boldsymbol{\lambda}\cdot\langle b\rangle)$,

$(\uparrow_\phi\downarrow 5)$ моделирование синхронной дивергенции Δ-символом:

$\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}\ \&\ \boldsymbol{\kappa}`=\underline{\boldsymbol{\lambda}`}\in(A\cap\underline{B})^{\infty}\ \vdash \boldsymbol{\mu}\cdot\langle\Delta\rangle\in(\boldsymbol{\kappa}\cdot\boldsymbol{\kappa}`)\uparrow\downarrow(\boldsymbol{\lambda}\cdot\boldsymbol{\lambda}`)$,

(|ϕ|6) индуктивный предел: композиция ϕ-трасс, хотя бы одна из которых бесконечна, содержит все ϕ-трассы, которые определяются бесконечной последовательностью применения правил (|ϕ|0÷4) к ϕ-трассам-операндам, причём в этой последовательности используется каждый из символов каждой ϕ-трассы-операнда.

□

Правила (|ϕ|1) и (|ϕ|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 $a\in A\setminus B$ и $b\in B\setminus A$ композиция $\langle a\rangle\wr\langle b\rangle=\{\langle a,b\rangle,\langle b,a\rangle\}$. Композиция (как множество) пуста, если операнды не компонуемы.

Аналогично один операнд $\boldsymbol{\kappa}$ и результирующая ϕ-трасса из композиционного множества $\boldsymbol{\mu}\in\boldsymbol{\kappa}\wr\boldsymbol{\lambda}$, вообще говоря, неоднозначно определяют другой операнд $\boldsymbol{\lambda}$. Например, для синхронного базового символа $z\in A\cap B$ и асинхронного базового символа $b\in A\setminus B$ композиционная ϕ-трасса $\langle b\rangle\in\langle z\rangle\wr\langle b,z\rangle$ и $\langle b\rangle\in\langle z\rangle\wr\langle z,b\rangle$.

Префикс композиционной ϕ-трассы $\boldsymbol{\mu}`\leq\boldsymbol{\mu}\in\boldsymbol{\kappa}\wr\boldsymbol{\lambda}$ принадлежит композиции некоторых префиксов операндов $\boldsymbol{\kappa}`\leq\boldsymbol{\kappa}$, $\boldsymbol{\lambda}`\leq\boldsymbol{\lambda}$, $\boldsymbol{\mu}`\in\boldsymbol{\kappa}`\wr\boldsymbol{\lambda}`$. 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 $\boldsymbol{\mu}`\in\boldsymbol{\kappa}`\wr\boldsymbol{\lambda}`$ является префиксом некоторой ϕ-трассы из композиции операндов $\boldsymbol{\mu}`\leq\boldsymbol{\mu}\in\boldsymbol{\kappa}\wr\boldsymbol{\lambda}$.

Если операнды $\boldsymbol{\kappa}$, $\boldsymbol{\lambda}$ и композиционная ϕ-трасса $\boldsymbol{\mu}\in\boldsymbol{\kappa}\wr\boldsymbol{\lambda}$ фиксированы, то они однозначно определяют последовательность применяемых правил

$(|_\phi|0\div4)$. Если хотя бы один из операндов бесконечен, то применяется правило индуктивного предела $(|_\phi|6)$ и, возможно, правило моделирования дивергенции Δ-символом $(|_\phi|5)$.

Если $\mu\in\kappa|||\lambda$ и ϕ-трассы представимы в виде $\kappa=\kappa`\cdot\kappa``$, $\lambda=\lambda`\cdot\lambda``$, $\mu=\mu`\cdot\mu``$ и префикс результата принадлежит композиции префиксов операндов $\mu`\in\kappa`|||\lambda`$, то постфикс результата принадлежит композиции постфиксов операндов $\mu``\in\kappa``|||\lambda``$.

Лемма 50: Композиция ϕ-трасс коммутативна:

$\forall A,B\subseteq Z^{\#}\ \forall\kappa\in\Phi^{\omega}(A)\ \forall\lambda\in\Phi^{\omega}(B)\ \kappa|||\lambda=\lambda|||\kappa$.

□524

Композиция маршрутов

Напомним, что композиция LTS без θ-переходов определяется следующим образом: Пусть $A,B\subseteq Z^{\#}$, $\kappa\in LTS(A_\gamma)$, $\lambda\in LTS(B_\gamma)$. Тогда $\kappa|||\lambda=LTS(V_\kappa\times V_\lambda,(A|||B)_\gamma,E,k_0l_0)$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall k\in V_\kappa\ \forall l\in V_\lambda$

$(|||1)\quad z\in A_{\gamma\tau}\setminus\underline{B}\ \&\ k\text{—}z\to k` \quad\vdash kl\text{—}z\to k`l;$

$(|||2)\quad z\in B_{\gamma\tau}\setminus\underline{A}\ \&\ l\text{—}z\to l` \quad\vdash kl\text{—}z\to kl`;$

$(|||3)\quad z\in A\cap\underline{B}\ \&\ k\text{—}z\to k`\ \&\ l\text{—}\underline{z}\to l` \quad\vdash kl\text{—}\tau\to k`l`.$

Определение 108: 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 $(||1 \div 3)$ к фиксированным маршрутам-операндам K и L. Если композиционный маршрут M бесконечен, то эта последовательность также бесконечная.

Обратно, при заданных маршрутах-операндах K и L маршрут $M \in K||L$ однозначно определяет последовательность применения правил $(||1 \div 3)$ при композиции. Исключение составляет случай, когда каждый из компонуемых префиксов 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:

$M \in K \wr\!\rfloor L \;\&\; M=M_1 \cdot M_2 \;\&\; K=K_1 \cdot K_2 \;\&\; L=L_1 \cdot L_2 \;\&\; M_1 \in K_1 \wr\!\rfloor L_1 \Rightarrow M_2 \in K_2 \wr\!\rfloor L_2.$

Покажем как связаны композиция маршрутов и композиция ϕ-трасс этих маршрутов.

Лемма 51: $\forall \boldsymbol{K}, \boldsymbol{L} \in \boldsymbol{LTS}_\gamma \;\; \forall K \in \boldsymbol{R}^\omega(\boldsymbol{K}) \;\; \forall L \in \boldsymbol{R}^\omega(\boldsymbol{L})$

1) $\forall M \in K \wr\!\rfloor L \;\; \forall \boldsymbol{\mu} \in \boldsymbol{\Phi}^\omega(M) \;\; \exists \boldsymbol{\kappa} \in \boldsymbol{\Phi}^\omega(K) \;\; \exists \boldsymbol{\lambda} \in \boldsymbol{\Phi}^\omega(L) \;\; \boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\!\rfloor \boldsymbol{\lambda}$,

2) $\forall \boldsymbol{\kappa} \in \boldsymbol{\Phi}^\omega(K) \;\; \forall \boldsymbol{\lambda} \in \boldsymbol{\Phi}^\omega(L) \;\; \forall \boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\!\rfloor \boldsymbol{\lambda} \;\; \exists M \in K \wr\!\rfloor L \;\; \boldsymbol{\mu} \in \boldsymbol{\Phi}^\omega(M)$.

□524

Аддитивность ϕ-трасс относительно композиции

Теорема 28: Условие Монотонность 3: выполнено: ϕ-трассы аддитивны относительно композиции:

$\forall \boldsymbol{K}, \boldsymbol{L} \in \boldsymbol{LTS}_\gamma \;\; \boldsymbol{\Phi}^\omega(\boldsymbol{K} \wr\!\rfloor \boldsymbol{L}) = \cup(\boldsymbol{\Phi}^\omega(\boldsymbol{K}) \wr\!\rfloor \boldsymbol{\Phi}^\omega(\boldsymbol{L}))$.

□530

6.4. Объединение ϕ-трасс конформных реализаций

Структура раздела:

1. Мажорирование ϕ-трасс как равенство
2. Конформность и мажорантность объединения
3. Существование преобразования

В этом разделе в качестве монотонной ϕ-спецификации мы рассмотрим множество всех конформных ϕ-трасс, то есть объединение конформных ϕ-реализаций. Соответственно, в качестве монотонной LTS-спецификации будем рассматривать (любую) LTS-модель, множество ϕ-трасс которой является монотонной ϕ-спецификацией. Такая ϕ-спецификация конформна и является наибольшей (по вложенности) монотонной ϕ-спецификацией. Заметим, что,

хотя класс конформных LTS-реализаций не является множеством, класс конформных ϕ-моделей является множеством как подкласс множества всех ϕ-моделей в данном алфавите.

Мы определим мажорирование ϕ-трасс «один ко многим» как принадлежность. Тогда мажорирование «один к одному» означает равенство, а «многие к многим» – вложенность. При таком определении генеративность, композиционность и рефлексивность (на классе всех LTS-моделей) ϕ-мажорирования доказываются тривиально. Также легко доказываются конформность и мажорантность преобразования (на классе всех LTS-моделей).

Заметим, что в этом разделе мы могли бы не использовать то $\mathfrak{R}$-мажорирование, которое определено в подразделе 6.2.1 и которое эквивалентно конформности. Нам было бы достаточно определить $\mathfrak{R}$-мажорирование ***F***-трасс «один к одному» как равенство, и распространить его на случай «один ко многим» как принадлежность и на случай «многие ко многим» как вложенность. По Лемме 16, вложенность влечёт конформность (условие Монотонность 1:), хотя обратное, конечно, не верно. Генеративность ϕ-мажорирования очевидна: вложенность множеств ϕ-трасс влечёт вложенность множеств порождаемых ими ***F***-трасс: $\mathbf{I}\subseteq\mathbf{\Sigma} \Rightarrow \cup_{\circ}\xi(\mathbf{I})\subseteq\cup_{\circ}\xi(\mathbf{\Sigma})$.

6.4.1. Мажорирование ϕ-трасс как равенство

Определение 109: Будем говорить, что ϕ-трасса мажорируется множеством ϕ-трасс, если эта ϕ-трасса принадлежит этому множеству:

для $\mu\in\mathbf{\Phi}^{\omega}(L)$ и $\mathbf{\Sigma}\subseteq\mathbf{\Phi}^{\omega}(L)$: $\mu\preccurlyeq\mathbf{\Sigma} =_{def} \mu\in\mathbf{\Sigma}$.

□

Теорема 29: ϕ-мажорирование по Определению 109 генеративно (Монотонность 4:), композиционно (Монотонность 5:) и рефлексивно (Монотонность 8:) на классе всех LTS-моделей.

□531

6.4.2. Конформность и мажорантность объединения

Определение 110: Определим семейство преобразований ***Conf***, которое для каждой $\mathfrak{R}$-семантики ставит в соответствие ф-спецификации $\mathbf{\Sigma}$ такую ф-спецификацию $\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{\Sigma})$, которая совпадает объединением ф-реализаций, конформных исходной ф-спецификации $\mathbf{\Sigma}$. Соответственно, определим семейство преобразований ***Conf***, которое для каждой $\mathfrak{R}$-семантики ставит в соответствие LTS-спецификации $\mathbf{S}$ такуюLTS-спецификацию $\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S})$, множество ф-трасс которой совпадает с объединением множеств ф-трасс всех LTS-реализаций, конформных исходной LTS-модели $\mathbf{S}$:

$\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{\Sigma}) = \cup\circ\mathfrak{I}_{\mathfrak{R}}(\mathbf{\Sigma})$, $\mathbf{\Phi}^{\omega}\circ\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S}) = \{\mathbf{\mu} \mid \exists \mathbf{I} \in \mathfrak{I}_{\mathfrak{R}}(\mathbf{S})\ \ \mathbf{\mu} \in \mathbf{\Phi}^{\omega}(\mathbf{I})\}$.

□

Лемма 52: Если преобразование $\boldsymbol{Conf}_{\mathfrak{R}}$ существует, то оно конформно (Монотонность 6:) и мажорантно (Монотонность 7:) на классе всех LTS-моделей.

□531

6.4.3. Существование преобразования

Теперь нам нужно доказать, что для каждой LTS-спецификации $\mathbf{S}$ существует LTS $\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S})$, множество ф-трасс которой совпадает с объединением ф-моделей, конформных $\mathbf{S}$. Для этого достаточно следующей Леммы.

Лемма 53: Объединение множества ф-моделей является ф-моделью.

□532

Таким образом, для ф-спецификации $\mathbf{\Sigma}$ объединение конформных ф-моделей $\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{\Sigma})$ является монотонной (и левомонотонной) ф-

спецификацией. Отсюда следует, что семейство преобразований ***Conf*** существует, если определить $\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S}) = \boldsymbol{\Phi 2L} \circ \boldsymbol{Conf}_{\mathfrak{R}} \circ \boldsymbol{\Phi}^{\omega}(\mathbf{S})$.

Теорема 30: Объединение $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ множества конформных ϕ-моделей является предельной (лево-)монотонной ϕ-моделью.

□533

Определение 111: Объединение $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ множества ϕ-моделей, конформных ϕ-спецификации $\boldsymbol{\Sigma}$, будем называть наибольшей монотонной ϕ-моделью. Соответственно, LTS-спецификацию $\boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S})$, множество ϕ-трасс которой совпадает с наибольшей монотонной ϕ-моделью, будем называть наибольшей монотонной LTS-моделью для LTS-спецификации $\mathbf{S}$.

□

6.5. Мажорирование ϕ-трасс

Структура раздела:

1. Определение мажорирования ϕ-трасс
2. Рефлексивность и транзитивность ϕ-мажорирования
3. Генеративность ϕ-мажорирования
4. Композиционность ϕ-мажорирования

Определение монотонного преобразования как наибольшей монотонной модели не оптимально, поскольку, как мы увидим ниже, для монотонности вовсе не требуется иметь все ϕ-трассы конформных реализаций. В этом разделе мы определим мажорирование ϕ-трасс «один к одному», отличное от равенства ϕ-трасс, но обладающее всеми свойствами, требуемыми общей теорией монотонности (генеративность и композиционность). На основе такого ϕ-мажорирования мы рассмотрим в следующих разделах монотонные ϕ-

спецификации, которые меньше (по вложенности), чем наибольшая монотонная ϕ-модель.

6.5.1. Определение мажорирования ϕ-трасс

Для общей теории монотонности требуется мажорирование ϕ-трасс как отношение «один ко многим». Мы определим ϕ-мажорирование как отношение «один к одному» и распространим его на случай «один ко многим».

Сначала определим мажорирование базовых символов (внешних действий, дивергенции и разрушения) и ϕ-символов. Для базовых символов a и b мажорирование $a \preccurlyeq b$ означает их равенство $a=b$. Для ϕ-символов a и b стабильных состояний s и t, соответственно, $a \preccurlyeq b$ означает: 1) если в s нет перехода по внешнему действию 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 внешнему действию 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 внешних действиях, а не просто о разрушающих. Если из состояния s есть переход $s \text{—} z \rightarrow s`$ по внешнему действию $z$, ведущий в опасное постсостояние $s` = \langle\gamma\rangle\Rightarrow$, в котором, тем не менее, нет γ-перехода $s` \text{—} \gamma \nrightarrow$, то действие z не попадает в ***gamma***-множество ϕ-символа состояния s.

Определение 112: Мажорирование символов. Определим отношение мажорирования базовых символов и ϕ-символов, а также отношение строго мажорирования ϕ-символов:

$\forall a,b \in L_{\Delta\gamma} \qquad a \preccurlyeq b =_{def} a=b;$

$\forall a,b\in\Phi(L)\quad a\preccurlyeq b =_{def} a_r \subseteq b_r\cup b_g \;\&\; a_g \subseteq b_g,$

$a\prec b =_{def} a\preccurlyeq b \;\&\; a\neq b.$

□

Мажорирование ϕ-трасс имеет две разновидности:

- α-мажорирование – поэлементное мажорирование символов для ϕ-трасс одной длины,
- γ-мажорирование – такое же поэлементное мажорирование префиксов ϕ-трасс и продолжение мажорирующего префикса разрушением.

Определение 113: Мажорирование ϕ-трасс.

- Определим отношение «один к одному» мажорирования ϕ-трасс:

  $\forall\boldsymbol{\mu},\boldsymbol{\sigma}\in\boldsymbol{\Phi}^{\omega}(L)\quad \boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma} =_{def} \boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma} \vee \boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\sigma}$, где:

  A)α-мажорирование: ϕ-трассы имеют одну длину и каждый символ мажорируемой ϕ-трассы мажорируется соответствующим символом мажорирующей ϕ-трассы:

  $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma} =_{def} |\boldsymbol{\mu}|=|\boldsymbol{\sigma}| \;\&\; \forall i\in[1..|\boldsymbol{\mu}|]\ \boldsymbol{\mu}(i)\preccurlyeq\boldsymbol{\sigma}(i);$

  B)γ-мажорирование: мажорирующая ϕ-трасса продолжается разрушением после префикса, α-мажорирующего префикс мажорируемой ϕ-трассы:

  $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\sigma} =_{def} \exists\boldsymbol{\mu}_1\ \exists\boldsymbol{\sigma}_1\ \boldsymbol{\mu}_1\leq\boldsymbol{\mu} \;\&\; \boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\sigma}_1 \;\&\; \boldsymbol{\sigma}=\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle.$

- Строгое мажорирование:

  $\boldsymbol{\mu}\prec\boldsymbol{\sigma} =_{def} \boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma} \;\&\; \boldsymbol{\mu}\neq\boldsymbol{\sigma},$

- Распространим мажорирование ϕ-трасс на случай «один ко многим»:

  $\forall\boldsymbol{\mu}\in\boldsymbol{\Phi}^{\omega}(L)\quad \forall\boldsymbol{\Sigma}\subseteq\boldsymbol{\Phi}^{\omega}(L)\quad \boldsymbol{\mu}\preccurlyeq\boldsymbol{\Sigma} =_{def} \exists\boldsymbol{\sigma}\in\boldsymbol{\Sigma}\ \boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}.$

□

Заметим, что $\mu \preccurlyeq^{\gamma} \sigma_1 \cdot \langle\gamma\rangle$ не исключает $\mu \preccurlyeq^{\alpha} \sigma_1 \cdot \langle\gamma\rangle$ или $\mu \preccurlyeq^{\alpha} \sigma_1$.

Как это было в общей теории монотонности отношение «один ко многим» распространяется на случай «многие ко многим»:

$\forall M, \Sigma \subseteq \Phi^{\omega}(L) \quad M \preccurlyeq \Sigma =_{def} \forall \mu \in M \ \ \mu \preccurlyeq \Sigma$.

6.5.2. Рефлексивность и транзитивность ϕ-мажорирования

Для общей теории монотонности требуется рефлексивность ϕ-мажорирования «многие ко многим» на выбранном классе LTS-спецификаций $\mathfrak{C}$. Мы покажем, что ϕ-мажорирование «один к одному» является частичным порядком (рефлексивно, транзитивно и антисимметрично) для любых LTS-спецификаций. Отсюда непосредственно следует, что ϕ-мажорирование «многие ко многим» является предпорядком (рефлексивно и транзитивно), хотя, вообще говоря, не является частичным порядком (может не выполняться свойство антисимметричности).

Теорема 31: Условие Монотонность 8: выполнено: Мажорирование ϕ-трасс «один к одному» является частичным порядком (рефлексивно, транзитивно и антисимметрично), а мажорирование ϕ-трасс «многие ко многим» является предпорядком (рефлексивно и транзитивно).

□533

6.5.3. Генеративность ϕ-мажорирования

Теорема 32: Условие Монотонность 4: выполнено: ϕ-мажорирование генеративно:

$\forall I, S \in LTS(L_{\gamma}) \quad \Phi^{\omega}(I) \preccurlyeq \Phi^{\omega}(S) \ \Rightarrow \ \cup \circ \xi \circ \Phi^{\omega}(I) \preccurlyeq_{\mathfrak{R}} \cup \circ \xi \circ \Phi^{\omega}(S)$.

□536

Из доказательства этой теоремы видно, что на самом деле ϕ-мажорирование влечёт более сильное отношение ***F***-трасс, чем $\mathfrak{R}$-мажорирование по Определению 83, а именно, отсутствует $\mathfrak{R}$-мажорирование вида $\preccurlyeq_{\mathfrak{R}}^{\Delta\gamma}$, а в $\mathfrak{R}$-мажорировании вида $\preccurlyeq_{\mathfrak{R}}^{z}$ вместо условия $\forall P\in\mathfrak{R}\ (z\in P \Rightarrow \exists t\in P\ \pi\cdot\langle t,\gamma\rangle\in\Sigma)$ используется более сильное условие $\pi\cdot\langle z,\gamma\rangle\in\Sigma$. С точки зрения общей теории монотонности мы могли бы использовать это более сильное отношение $\mathfrak{R}$-мажорирования, но оно не эквивалентно конформности.

6.5.4. Композиционность ϕ-мажорирования

Композиционность ϕ-мажорирования мы докажем на классе всех LTS-спецификаций, из чего будет следовать композиционность ϕ-мажорирования на любом его подклассе. Более того, мы покажем композиционность ϕ-мажорирования слева (при замене одного, для определённости, левого операнда на его мажоранту), а из неё уже выведем полную композиционность ϕ-мажорирования (при замене обоих операндов на их мажоранты).

Теорема 33: ϕ-мажорирование композиционно слева на классе всех LTS-спецификаций:

$\forall \mathbf{I_1}, \mathbf{I_2}, \mathbf{S_1}\in \boldsymbol{LTS}_\gamma\ \ \boldsymbol{\Phi}^\omega(\mathbf{I_1})\preccurlyeq\boldsymbol{\Phi}^\omega(\mathbf{S_1})$

$\Rightarrow\quad \cup(\boldsymbol{\Phi}^\omega(\mathbf{I_1})\between\boldsymbol{\Phi}^\omega(\mathbf{I_2}))\preccurlyeq\cup(\boldsymbol{\Phi}^\omega(\mathbf{S_1})\between\boldsymbol{\Phi}^\omega(\mathbf{I_2}))$.

□539

Теорема 34: Условие Монотонность 5: выполнено: ϕ-мажорирование композиционно на классе всех LTS-спецификаций:

$\forall \mathbf{I}_1, \mathbf{I}_2, \mathbf{S}_1, \mathbf{S}_2 \in \boldsymbol{LTS}_\gamma$

$$\boldsymbol{\Phi}^\omega(\mathbf{I}_1) \preccurlyeq \boldsymbol{\Phi}^\omega(\mathbf{S}_1) \quad \& \quad \boldsymbol{\Phi}^\omega(\mathbf{I}_2) \preccurlyeq \boldsymbol{\Phi}^\omega(\mathbf{S}_2)$$

$$\Rightarrow \quad \cup(\boldsymbol{\Phi}^\omega(\mathbf{I}_1) \upharpoonright\downarrow \boldsymbol{\Phi}^\omega(\mathbf{I}_2)) \preccurlyeq \cup(\boldsymbol{\Phi}^\omega(\mathbf{S}_1) \upharpoonright\downarrow \boldsymbol{\Phi}^\omega(\mathbf{S}_2)).$$

□544

6.6. Монотонные модели

Структура раздела:

1. Финальность
2. Однородность
3. Сингулярность

Мажорирование ϕ-трасс, определённое в предыдущем разделе, позволяет в качестве монотонной спецификации использовать не всё множество конформных ϕ-трасс, то есть наибольшую монотонную ϕ-модель $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$, а некоторые его подмножества. Поскольку выше доказано выполнение условий монотонности 1÷5 и 8, для (лево-)монотонности ϕ-модели требуется выполнение условия 6 – конформность (исходной спецификации), и условия 7 – мажорантность (мажорирование конформных ϕ-трасс). Любая подмодель наибольшей монотонной ϕ-модели конформна (Лемма 16). Поэтому нам нужно искать такие подмодели, которые мажорируют все конформные ϕ-трассы. Также, поскольку мажорирование ϕ-трасс рефлексивно (условие монотонности 8), любая монотонная модель, удовлетворяющая условиям монотонности 6 и 7, будет также левомонотонной. Поэтому в дальнейшем мы не будем делать различия между левомонотонными и монотонными моделями.

Теорема 35: ϕ-модель, являющаяся подмножеством монотонной ϕ-модели и мажорирующая её, монотонная.

□544

Идеальным решением было бы нахождение наименьшей или хотя бы минимальной (по вложенности) монотонной модели. Однако общая теория монотонности не даёт ответа на вопрос о *необходимых* условиях монотонности, определяя только достаточные условия. Поэтому приходится говорить о минимальной или наименьшей *мажорантной* подмодели наибольшей монотонной модели. К сожалению, как будет показано ниже, такие модели не всегда существуют, если не налагать на спецификацию специальных ограничений. Мы рассмотрим три вложенных мажорантных подмодели: наибольшую финальную ⊇ наибольшую однородную ⊇ наибольшую сингулярную. При этом последняя подмодель в общем случае может не существовать. Для её существования мы рассмотрим специальные ограничения на спецификацию (в следующем разделе). Также мы исследуем вопрос о связи наименьшей и минимальных мажорантных подмоделей с наибольшей сингулярной подмоделью, когда эти подмодели существуют.

Там, где не оговорено противное, мы будем предполагать, что задана некоторая $\mathfrak{R}$-семантика и ϕ-спецификация $\mathbf{\Sigma}$. Поэтому вместо $\boldsymbol{Conf}_{\mathfrak{R}}$ будем писать просто $\boldsymbol{Conf}$.

Мы будем часто использовать понятие «безопасная $\mathfrak{R}$-трасса», имея в виду, что она безопасна в исходной спецификации. В то же время, поскольку спецификация и любая монотонная модель конформны друг другу, из Теорема 18: следует, что они имеют одинаковые множества безопасных $\mathfrak{R}$-трасс.

Прежде всего, отметим, что мы можем сделать все разрушающие действия непосредственно разрушающими и получить мажорирующую подмодель наибольшей монотонной ϕ-модели $\boldsymbol{Conf}(\mathbf{\Sigma})$. Для этого нужно в каждой ϕ-трассе в каждом её ϕ-символе добавить в ***gamma***-множество каждое внешнее

действие, разрушающее (не обязательно непосредственно) после префикса ϕ-трассы, заканчивающегося этим ϕ-символом. Это значит применить одновременно все возможные операции вида: $\boldsymbol{\mu}\cdot\langle o\rangle\cdot\boldsymbol{\lambda}\to\boldsymbol{\mu}\cdot\langle(o_r, o_g\cup U)\rangle\cdot\boldsymbol{\lambda}$ при условии $U=\{z\in L \mid \boldsymbol{\mu}\cdot\langle o, z, \gamma\rangle\in \boldsymbol{Conf}(\boldsymbol{\Sigma})\}$. Действительно, мы получим *модифицированную* ϕ-модель, очевидно, имеющую то же множество $\mathfrak{R}$-трасс (они не зависят от ***gamma***-множеств), но другое множество ϕ-трасс, мажорирующее множество ϕ-трасс исходной ϕ-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$. Последнее объясняется мажорированием соответствующих ϕ-символов соответствующих ϕ-трасс: $(o_r, o_g)\preccurlyeq(o_r, o_g\cup U)$. Заметим, что модифицированная ϕ-модель, очевидно, является подмоделью ϕ-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$.

Если такое преобразование применять к канонической наибольшей монотонной LTS-модели $\boldsymbol{Ф2L}\circ\boldsymbol{Conf}(\boldsymbol{\Sigma})$, то оно сводится к замене каждого τ-перехода в состояние с γ-петлёй $\boldsymbol{\mu}\text{—}\tau\to\mathbf{t}\text{—}\gamma\to\mathbf{t}$ на γ-петлю $\boldsymbol{\mu}\text{—}\gamma\to\boldsymbol{\mu}$. Это эквивалентно использованию вместо преобразования ***Ф2L*** следующего *модифицированного* преобразования (состояние $\mathbf{t}$ теперь не требуется):

(1) $\boldsymbol{\mu}\cdot\langle z\rangle\in V_{\boldsymbol{s}}$ $\vdash$ $\boldsymbol{\mu}\text{—}z\to\boldsymbol{\mu}\cdot\langle z\rangle$;

(2) $\boldsymbol{\mu}\cdot\langle o\rangle\in V_{\boldsymbol{s}}$ $\vdash$ $\boldsymbol{\mu}\text{—}\tau\to\boldsymbol{\mu}\cdot\langle o\rangle$;

(3) $\boldsymbol{\mu}\in V_{\boldsymbol{s}}$ & $\boldsymbol{\mu}\cdot\langle\Delta\rangle\in\boldsymbol{\Sigma}$ $\vdash$ $\boldsymbol{\mu}\text{—}\tau\to\boldsymbol{\mu}$;

(4) $\boldsymbol{\mu}\in V_{\boldsymbol{s}}$ & $\boldsymbol{\mu}\cdot\langle\gamma\rangle\in\boldsymbol{\Sigma}$ $\vdash$ $\boldsymbol{\mu}\text{—}\tau\to\boldsymbol{\mu}$.

Далее, если не оговорено противное, под конформными ϕ-трассами мы будем понимать конформные ϕ-трассы, для которых после каждого префикса каждое внешнее действие, разрушающее после этого префикса, является непосредственно разрушающим. Иными словами, это будут конформные ϕ-трассы *модифицированной* ϕ-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$ или, что то же самое,

конформные ϕ-трассы канонической LTS-модели $\boldsymbol{\Phi 2L}\circ\boldsymbol{Conf}(\boldsymbol{\Sigma})$ с использованием *модифицированного* преобразования $\boldsymbol{\Phi 2L}$. Также легко увидеть, что такая модификация сохраняет предельность ϕ-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$.

6.6.1. Финальность

В этом подразделе мы рассмотрим первую монотонную подмодель, которую будем называть наибольшей финальной ϕ-моделью. Идея финальности основана на том, что конформная ϕ-трасса, продолжаемая разрушением, может иметь и любое другое продолжение, но все эти другие продолжения можно удалить из монотонной ϕ-модели, не нарушая монотонности. Также для конформной ϕ-трассы, продолжаемой дивергенцией, из всех других продолжений достаточно оставить только продолжения разрушающими действиями и ϕ-символами.

В канонической наибольшей монотонной LTS-модели $\boldsymbol{\Phi 2L}\circ\boldsymbol{Conf}(\boldsymbol{\Sigma})$ удалению нефинальных ϕ-трасс соответствует (Рис.68):

1) для состояния $\boldsymbol{\mu}$, в котором определена γ-петля $\boldsymbol{\mu}\text{—}\gamma\text{→}\boldsymbol{\mu}$, удаление всех остальных переходов $\boldsymbol{\mu}\text{—}u\text{→}$, где $u\neq\gamma$;
2) для состояния $\boldsymbol{\mu}$, в котором нет γ-петли, но определена τ-петля $\boldsymbol{\mu}\text{—}\tau\text{→}\boldsymbol{\mu}$, удаление переходов по внешним действиям $\boldsymbol{\mu}\text{—}z\text{→}\boldsymbol{\mu}\cdot\langle z\rangle$, где $z\in L$, ведущих в постсостояния без γ-петель $\boldsymbol{\mu}\cdot\langle z\rangle\text{—}\gamma\text{↛}$.

Удаление нефинальных ϕ-трасс из $\Phi 2L_{\circ}Conf(\Sigma)$	
$\mu\cdot\langle\gamma\rangle\in Conf(\Sigma)$	$\mu\cdot\langle\gamma\rangle\notin Conf(\Sigma)$ & $\mu\cdot\langle\Delta\rangle\in Conf(\Sigma)$

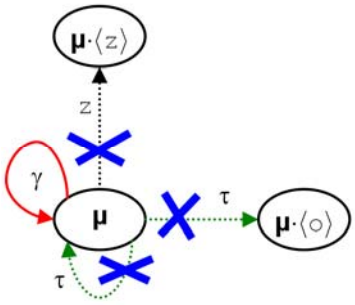
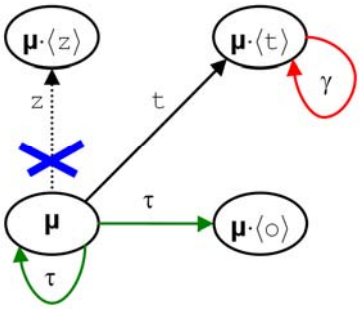

	

Рис.68.	

Определение 114:

- Конформную ϕ-трассу $\sigma\in Conf(\Sigma)$ будем называть *безопасной*, если она генерирует (через ξ-оператор) хотя бы одну безопасную $\mathfrak{R}$-трассу: $\xi(\sigma)\cap Safe(\Sigma)\neq\varnothing$.
- Внешнее действие или $\mathfrak{R}$-кнопку будем называть *опасными* после безопасной ϕ-трассы σ, если они опасны после каждой безопасной (в исходной спецификации) $\mathfrak{R}$-трассы, генерируемой ϕ-трассой σ.

□

Очевидно, что внешнее действие опасно после ϕ-трассы тогда и только тогда, когда оно разрешается только такими $\mathfrak{R}$-кнопками, которые опасны после ϕ-трассы.

Определение 115:

- ϕ-трассу $\boldsymbol{\sigma}$ будем называть *финальной*, если
  a) $\langle\gamma\rangle\in\boldsymbol{\Sigma}$ & $(\boldsymbol{\sigma}=\epsilon \vee \boldsymbol{\sigma}=\langle\gamma\rangle)$,
  b) или для некоторой безопасной ϕ-трассы $\boldsymbol{\mu}$ имеет место один из следующих случаев:
     - $\boldsymbol{\sigma}=\boldsymbol{\mu}$,
     - $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle\Delta\rangle$, если такое продолжение согласовано ($\boldsymbol{postf}(\boldsymbol{\mu})=\epsilon$), а все $\mathfrak{R}$-кнопки опасны после $\boldsymbol{\mu}$,
     - $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z\rangle$ или $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$, если такое продолжение согласовано $(\forall o\in \boldsymbol{Im}\circ\boldsymbol{postf}(\boldsymbol{\mu})\ z\notin o_r)$, а действие $z\in L$ опасно после $\boldsymbol{\mu}$.
- Множество финальных ϕ-трасс обозначим $\boldsymbol{Final}(\boldsymbol{\Sigma})$, а его ***d***-замыкание будем называть *наибольшей финальной ϕ-моделью*. LTS, множество ϕ-трасс которой равно наибольшей финальной ϕ-модели, будем называть *наибольшей финальной LTS*.
- *Финальной ϕ-моделью* будем называть монотонную ϕ-модель, вложенную в наибольшую финальную ϕ-модель. LTS, множество ϕ-трасс которой является финальной ϕ-моделью, будем называть *финальной LTS*.

□

Если в спецификации есть трасса $\langle\gamma\rangle$, то безопасных ϕ-трасс нет, и единственными финальными ϕ-трассами являются пустая ϕ-трасса ϵ и γ-трасса $\langle\gamma\rangle$. Если в спецификации нет трассы $\langle\gamma\rangle$, то, по крайней мере, пустая ϕ-трасса безопасна.

Если все действия опасны после безопасной ϕ-трассы $\boldsymbol{\mu}$, то все непустые $\mathfrak{R}$-кнопки опасны после $\boldsymbol{\mu}$. В этом случае ϕ-трасса $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ финальна тогда и только тогда, когда нет пустой $\mathfrak{R}$-кнопки или она опасна после $\boldsymbol{\mu}$. Последнее

возможно только в том случае, когда каждая безопасная $\mathfrak{R}$-трасса, генерируемая $\boldsymbol{\mu}$, продолжается в спецификации дивергенцией.

Для того, чтобы название «наибольшая финальная ϕ-модель» было корректным, нужно, чтобы множество $\boldsymbol{d}_{\circ}\boldsymbol{Final}(\boldsymbol{\Sigma})$ было ϕ-моделью, что показывается ниже в Теорема 36:.

Лемма 54: Финальные ϕ-трассы конформны.

□545

Отсюда следует, что, при отсутствии в спецификации γ-трассы, финальная ϕ-трасса – это либо безопасная ϕ-трасса $\boldsymbol{\mu}$, либо её конформное продолжение дивергенцией $\boldsymbol{\mu}\cdot\langle\Delta\rangle$, либо её конформное продолжение действием и разрушением $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$, а также префикс последней ϕ-трассы $\boldsymbol{\mu}\cdot\langle z\rangle$.

Лемма 55: Необходимым и достаточным условием финальности конформной ϕ-трассы $\boldsymbol{\sigma}\in\boldsymbol{Conf}(\boldsymbol{\Sigma})$ является отсутствие отличных от неё самой конформных γ-ответвлений после меньшего её префикса: $\forall\boldsymbol{\mu}<\boldsymbol{\sigma}\ \ \boldsymbol{\mu}\cdot\langle\gamma\rangle=\boldsymbol{\sigma}\ \vee\ \boldsymbol{\mu}\cdot\langle\gamma\rangle\notin\boldsymbol{Conf}(\boldsymbol{\Sigma})$.

□547

Теорема 36: Множество $\boldsymbol{d}_{\circ}\boldsymbol{Final}(\boldsymbol{\Sigma})$ является монотонной предельной ϕ-моделью.

□549

6.6.2. Однородность

С точки зрения монотонности, наибольшая финальная ϕ-модель всё ещё «слишком велика» и может иметь строго меньшие (по вложенности) монотонные подмодели. В этом подразделе мы рассмотрим одну такую подмодель, которая состоит из всех однородных ϕ-трасс. Идея однородности основана на том, что действие, опасное после конформной ϕ-трассы, может не

быть разрушающим, но в монотонной ϕ-модели можно ограничиться только теми ϕ-трассами, в которых все опасные действия разрушающие. Напомним, что мы рассматриваем только такие монотонные модели, в которых все разрушающие действия являются непосредственно разрушающими. Пусть ϕ-трасса $\boldsymbol{\mu}$ продолжается ϕ-символом $\circ$. Действие, опасное после $\boldsymbol{\mu}\cdot\langle\circ\rangle$, может как входить, так и не входить в ***gamma***-множество $\circ_g$. Если всегда имеет место первый случай, будем говорить, что ϕ-символ $\circ$ однороден после ϕ-трассы $\boldsymbol{\mu}$.[30] Будем говорить, что ϕ-трасса однородна, если все её ϕ-символы однородны после соответствующих префиксов ϕ-трассы.

В канонической LTS-модели множества финальных ϕ-трасс $\boldsymbol{\Phi 2L\circ d\circ Final}(\boldsymbol{\Sigma})$ удалению неоднородных ϕ-трасс соответствует удаление всех τ-переходов $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle\circ\rangle$, ведущих в стабильные состояния с неоднородным ϕ-символом $\circ$ (Рис.69).

Однородные и неоднородные ϕ-символы

Однородный ϕ-символ a	Неоднородный ϕ-символ b

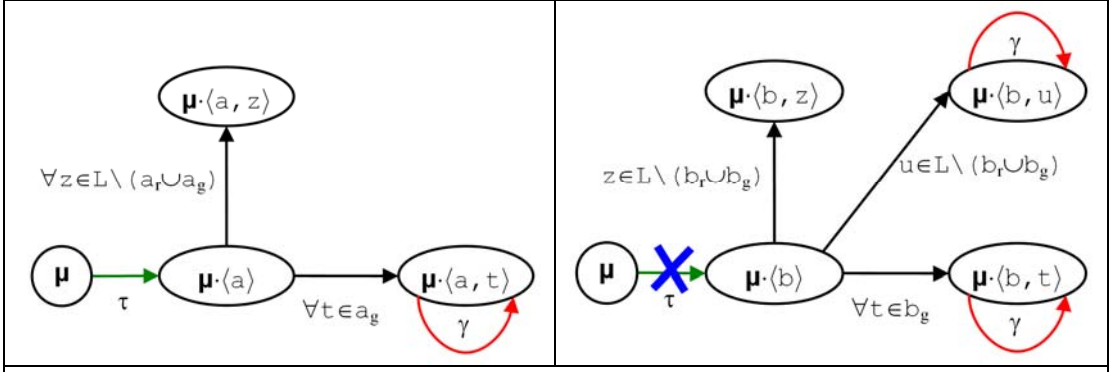


Рис.69.

[30] Для βγδ-семантики, где имеется $\mathfrak{R}$-кнопка для каждого стимула и общая $\mathfrak{R}$-кнопка для всех реакций, ***gamma***-множество однородного ϕ-символа либо не содержит реакций, либо содержит все реакции.

Определение 116:

- ϕ-символ $o \in \Phi(L)$ будем называть *однородным* после безопасной ϕ-трассы $\boldsymbol{\mu}$, если ϕ-трасса $\boldsymbol{\mu}\cdot\langle o\rangle$ безопасна, а каждое действие, опасное после $\boldsymbol{\mu}\cdot\langle o\rangle$, принадлежит ***gamma***-множеству o_g.
- Финальную ϕ-трассу будем называть *однородной*, если каждый её ϕ-символ однороден после непосредственно предшествующего ему префикса ϕ-трассы.
- Множество однородных ϕ-трасс обозначим $\boldsymbol{Uniform}(\boldsymbol{\Sigma})$, а его ***d***-замыкание будем называть *наибольшей однородной ϕ-моделью*. LTS, множество ϕ-трасс которой равно наибольшей однородной ϕ-модели, будем называть *наибольшей однородной LTS*.
- *Однородной ϕ-моделью* будем называть монотонную ϕ-модель, вложенную в наибольшую однородную ϕ-модель. LTS, множество ϕ-трасс которой является однородной ϕ-моделью, будем называть *однородной LTS*.

□

По определению, $\boldsymbol{Uniform}(\boldsymbol{\Sigma}) \subseteq \boldsymbol{Final}(\boldsymbol{\Sigma})$ и, следовательно, $\boldsymbol{d}\circ\boldsymbol{Uniform}(\boldsymbol{\Sigma}) \subseteq \boldsymbol{d}\circ\boldsymbol{Final}(\boldsymbol{\Sigma})$. Для того, чтобы название «наибольшая однородная ϕ-модель» было корректным, нужно, чтобы множество $\boldsymbol{d}\circ\boldsymbol{Uniform}(\boldsymbol{\Sigma})$ было ϕ-моделью, что показывается ниже в Лемме 59.

Определение 117: Будем говорить, что две финальные ϕ-трассы $\boldsymbol{\mu}$ и $\boldsymbol{\sigma}$ *похожи* и обозначать $\boldsymbol{\mu}\sim\boldsymbol{\sigma}$, если они отличаются только ***gamma***-множествами соответствующих ϕ-символов:

$$\boldsymbol{\mu}\sim\boldsymbol{\sigma} =_{def} |\boldsymbol{\mu}| = |\boldsymbol{\sigma}| \ \&\ \forall i \in [1..|\boldsymbol{\mu}|]$$
$$(\ \boldsymbol{\mu}(i) = \boldsymbol{\sigma}(i) \in L \vee \boldsymbol{\mu}(i), \boldsymbol{\sigma}(i) \in \Phi(L)\ \&\ \boldsymbol{\mu}(i)_r = \boldsymbol{\sigma}(i)_r\ ).$$

□

Отношение похожести, очевидно, является эквивалентностью (рефлексивно, симметрично и транзитивно).

Лемма 56: Похожие финальные ϕ-трассы генерируют одинаковые ***F***-трассы: $\boldsymbol{\mu}\sim\boldsymbol{\sigma} \Rightarrow \xi(\boldsymbol{\mu})=\xi(\boldsymbol{\sigma})$.

□551

Лемма 57: Похожие однородные ϕ-трассы совпадают.

□551

Лемма 58: Для любой финальной ϕ-трассы $\boldsymbol{\mu}$ найдётся (и единственная) похожая однородная ϕ-трасса $\boldsymbol{\sigma}$, причём $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$.

□552

Заметим, что, если финальная ϕ-трасса имеет вид $\boldsymbol{\mu}\cdot\langle\circ\rangle$, где $\circ$ ϕ-символ, и ϕ-трасса $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ также финальна, то похожая однородная ϕ-трасса имеет вид $\boldsymbol{\sigma}\cdot\langle(\varnothing,\mathtt{L})\rangle$, поскольку после $\boldsymbol{\mu}\cdot\langle\circ\rangle$ все действия опасны.

Лемма 59: ***d***-замыкание множества однородных ϕ-трасс является предельной ϕ-моделью, мажорирующей наибольшую финальную ϕ-модель.

□555

Теорема 37: Наибольшая однородная ϕ-модель является монотонной предельной ϕ-моделью.

□556

6.6.3. Сингулярность

Структура подраздела:

- Предоминанты ϕ-символа
- Сингулярные действия, ϕ-символы и ϕ-трассы
- Соотношение минимальных и сингулярных ϕ-моделей
- Соотношение наименьшей и сингулярной ϕ-моделей

Переход к наибольшей финальной ϕ-модели уменьшает длину конформных ϕ-трасс, поскольку при конформности ϕ-трассы $\boldsymbol{\mu}\cdot\langle\gamma\rangle$ удаляется

любое другое продолжение $\boldsymbol{\mu}\cdot\boldsymbol{\lambda}$ ($\boldsymbol{\lambda}\neq\epsilon$ & $\boldsymbol{\lambda}\neq\langle\gamma\rangle$), а при конформности ϕ-трассы $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ (и неконформности трассы $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$) удаляется любое продолжение $\boldsymbol{\mu}\cdot\langle z\rangle\cdot\boldsymbol{\lambda}$, начинающееся с неразрушающего действия z. Далее наибольшая однородная ϕ-модель максимизирует ***gamma***-множества ϕ-символов: ϕ-символы однородных ϕ-трасс имеют максимальные ***gamma***-множества среди всех похожих финальных ϕ-трасс, поскольку состоят из всех действий, опасных после соответствующего префикса ϕ-трассы. Следующий этап минимизации монотонной ϕ-модели – это максимизация ***ref***-множеств ϕ-символов. Этот этап будем называть *сингуляризацией* и рассмотрим его в данном подразделе.

Предоминанты ϕ-символа

Как показано в Лемме 57, похожие однородные ϕ-трассы совпадают: ***ref***-множества их ϕ-символов совпадают по определению похожести, а ***gamma***-множества совпадают из-за однородности. $\mathfrak{R}$-похожими ϕ-трассами назовём такие однородные ϕ-трассы, которые отличаются только ***ref***-множествами соответствующих ϕ-символов, причём эти ***ref***-множества, хотя и не одинаковы, но порождают одно и то же множество $\mathfrak{R}$-отказов. Сингуляризация сводится удалению $\mathfrak{R}$-похожих ϕ-трасс с меньшими ***ref***-множествами ϕ-символов. Мы удаляем ϕ-трассу $\boldsymbol{\mu}$, если остаётся $\mathfrak{R}$-похожая ϕ-трасса $\boldsymbol{\mu}`$, каждый ϕ-символ $\circ`$ которой имеет ***ref***-множество большее или равное ***ref***-множеству соответствующего ϕ-символа $\circ$ ϕ-трассы $\boldsymbol{\mu}$: $\circ_r\subseteq\circ`_r$. Поскольку в $\mathfrak{R}$-похожих ϕ-трассах соответствующие ϕ-символы имеют одинаковые ***gamma***-множества $\circ_g=\circ`_g$, мы имеем мажорирование $\circ_r\preccurlyeq\circ`_r$. Тем самым гарантируется мажорирование удаляемых ϕ-трасс остающимися ϕ-трассами.

Однако, если имеет место строгое вложение $\circ_r\subset\circ`_r$, то может исчезнуть конформное продолжение ϕ-трассы безопасным действием $z\in\circ`_r\setminus\circ_r$. Чтобы

этого не произошло, для каждого безопасного действия $z \in L \setminus o_r \setminus o_g$ должна оставаться какая-то другая $\mathfrak{R}$-похожая ϕ-трасса $\boldsymbol{\mu}_z$ такая, что действие z не принадлежит ***ref***-множеству соответствующего ϕ-символа o_z: $o_r \subseteq o_{zr}$ & $z \notin o_{zr}$. Фактически, для каждой удаляемой ϕ-трассы $\boldsymbol{\mu}$ должно оставаться некоторое семейство $\mathfrak{R}$-похожих ϕ-трасс, чтобы покрыть все такие ответвления безопасными действиями.

На уровне ϕ-символов это означает, что мы заменяем ϕ-символ o на некоторое семейство O мажорирующих его ϕ-символов, каждый из которых имеет то же ***gamma***-множество, не меньшее ***ref***-множество и то же множество порождаемых $\mathfrak{R}$-отказов. Такие ϕ-символы будем называть *доминирующими* над ϕ-символом o. Дополнительно требуется, чтобы для каждого действия z, не принадлежащего ***ref***- и ***gamma***-множествам ϕ-символа o, в семействе нашёлся ϕ-символ с тем же свойством: действие z не принадлежит его ***ref***- и ***gamma***-множествам. Семейство доминирующих ϕ-символов с таким дополнительным свойством будем называть *предоминантой* ϕ-символа o.

Определение 118:

- Для множества внешних действий $P \subseteq L$ обозначим множество вложенных в него $\mathfrak{R}$-отказов:

  $\mathfrak{R}(P) =_{def} \{Q \in \mathfrak{R} \mid Q \subseteq P\}$.

- Определим отношение *доминирования* ϕ-символов:

  $a \trianglelefteq b =_{def} a_g = b_g \;\&\; a_r \subseteq b_r \;\&\; \mathfrak{R}(a_r) = \mathfrak{R}(b_r)$.

- Множество всех ϕ-символов, доминирующих ϕ-символ a, обозначим:

  $a^{\triangle} =_{def} \{b \in \Phi(L) \mid a \trianglelefteq b\}$.

· Подмножество доминирующих ϕ-символов $B \subseteq a^{\triangle}$ будем называть *предоминантой* ϕ-символа a, если выполнено следующее условие: $\forall z \in L \setminus (a_r \cup a_g)\ \exists b \in B\ z \notin b_r$.

□

Очевидно, что отношение доминирования является (нестрогим) частичным порядком (рефлексивно, транзитивно и антисимметрично). Поэтому множество $a^{\triangle}$ означает верхний конус ϕ-символа a по этому отношению.

Заметим, что, вообще говоря, предоминанта ϕ-символа a определяется неоднозначно, то есть у ϕ-символа a может быть несколько предоминант. Само множество доминирующих ϕ-символов (верхний конус $a^{\triangle}$) является одной из таких предоминант. Нас будут интересовать, в первую очередь, те предоминанты, в которые не входит сам ϕ-символ, если такие есть.

В канонической LTS-модели замене ϕ-символа o на его предоминанту O соответствует:

1) замена стабильного состояния s с ϕ-символом o на множество состояний $s_1, s_2, \ldots$, соответствующих ϕ-символам $o_1, o_2, \ldots$ предоминанты O;
2) замена τ-перехода $t \text{—}\tau\rightarrow s$ на множество τ-переходов вида $t \text{—}\tau\rightarrow s_i$ во все добавляемые состояния $s_1, s_2, \ldots$;
3) для каждого действия $z \in o_g$ замена перехода $s \text{—}z\rightarrow v$ на множество переходов вида $s_i \text{—}z\rightarrow v$ из всех добавляемых состояний $s_1, s_2, \ldots$;
4) для каждого действия $z \in L \setminus o_r \setminus o_g$ замена перехода $s \text{—}z\rightarrow w$ на множество переходов вида $s_i \text{—}z\rightarrow w$ для всех тех добавляемых состояний s_i, для которых $z \notin o_{ir}$.

Поскольку все ϕ-символы, доминирующие ϕ-символ o, порождают то же множество $\mathfrak{R}$-отказов, что и сам ϕ-символ o, наибольшая однородная ϕ-модель вместе с каждым ϕ-символом o содержит все его доминирующие ϕ-символы, а значит все его предоминанты. В канонической LTS-модели это означает, что все добавляемые состояния $s_1, s_2, \ldots$, на самом деле, уже имеются. Иными словами, замена состояния s на состояния $s_1, s_2, \ldots$, на самом деле, означает лишь удаление самого состояния s, разумеется, при условии, что это не одно из состояний $s_1, s_2, \ldots$, то есть при условии, что ϕ-символ не входит в свою предоминанту $o \notin O$. Тогда мы получим меньшую монотонную ϕ-модель.

Сингулярные действия, ϕ-символы и ϕ-трассы

Проблема в том, что процесс замены ϕ-символов на их предоминанты в общем случае не всегда может быть завершён. Если ϕ-символ $o \notin O$, то мы можем удалить в канонической LTS-модели состояние s. После этого может оказаться, что для некоторого ϕ-символа $o_1 \in O$ также существует предоминанта O_1, которой он не принадлежит $o_1 \notin O_1$, и мы можем удалить состояние s_1. И так далее. Этот процесс останавливается, если через конечное число шагов мы получим только такие ϕ-символы, которые входят во все свои предоминанты и, следовательно, их уже нельзя удалять. Такие ϕ-символы будем называть *сингулярными ϕ-символами*, стабильные состояния LTS-модели, имеющие сингулярные ϕ-символы, будем называть *сингулярными состояниями*, а ϕ-трассы, все ϕ-символы которых сингулярны, будем называть *сингулярными ϕ-трассами*.

Можно дать независимое определение сингулярного ϕ-символа через понятие сингулярного действия. Действие z будем называть сингулярным для ϕ-символа o, если оно принадлежит ***ref***- или ***gamma***-множеству ϕ-символа, или не принадлежит этим множествам, но, будучи добавленным к ***ref***-множеству ϕ-символа o, меняет множество порождаемых им $\mathfrak{R}$-отказов. Последнее

эквивалентно тому, что действие принадлежит некоторому $\mathfrak{R}$-отказу, остальные действия которого принадлежат ***ref***-множеству ϕ-символа o. Тогда сингулярный ϕ-символ – это такой ϕ-символ, у которого есть не более одного несингулярного действия[31].

Если все действия сингулярны, множество доминирующих ϕ-символов состоит только из одного сингулярного ϕ-символа o; это множество $\{o\}$ является единственной предоминантой. Если есть одно несингулярное действие z, множество доминирующих ϕ-символов состоит из ϕ-символа o и ϕ-символа o_z, получаемого добавлением действия z в ***ref***-множество $o_z=(o_r\cup\{z\},o_g)$; имеются две предоминанты: $\{o\}$ и $\{o,o_z\}$.

В канонической LTS-модели сингулярное состояние – это такое стабильное состояние s, в котором определено не более одного перехода $s\text{—}z\rightarrow s`$, где $s`\text{—}\gamma\not\rightarrow$, удаление которого не меняет множество порождаемых этим состоянием $\mathfrak{R}$-отказов.

Определение 119:

- Действие z будем называть *сингулярным* для ϕ-символа o, если оно удовлетворяет следующему условию:
  $z\in o_r\cup o_g \vee z\notin o_r\cup o_g \;\&\; \exists P\in\mathfrak{R}\; z\in P \;\&\; P\setminus\{z\}\subseteq o_r$.
- ϕ-символ o будем называть *сингулярным*, если все действия, кроме, быть может, одного, сингулярны для этого ϕ-символа.

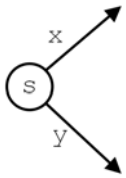
---

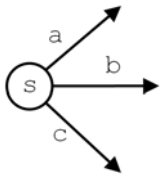
[31] Отсюда и происходит название «сингулярный». Также для βγδ-семантики, где имеется $\mathfrak{R}$-кнопка для каждого стимула и общая $\mathfrak{R}$-кнопка для всех реакций, ϕ-символ сингулярен, если объединение его ***ref***- и ***gamma***-множеств содержит все реакции, кроме, быть может, одной. В сингулярном состоянии LTS определено не более одного перехода по непосредственно не разрушающей реакции. Для однородного сингулярного ϕ-символа возможны три варианта: 1) ***gamma***-множество содержит все реакции, 2) ***ref***-множество содержит все реакции, 3) ***ref***-множество содержит все реакции, кроме одной, и эта реакция не принадлежит ***gamma***-множеству.

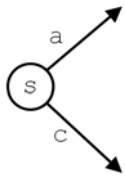
- *Сингулярным состоянием* LTS-модели будем называть её стабильное состояние, ϕ-символ которого сингулярен.
- *Сингулярной ϕ-трассой* будем называть такую однородную ϕ-трассу, каждый ϕ-символ которой сингулярен.
- Множество сингулярных ϕ-трасс обозначим $\boldsymbol{Sing}(\boldsymbol{\Sigma})$. Если ***d***-замыкание этого множества $\boldsymbol{d}\circ\boldsymbol{Sing}(\boldsymbol{\Sigma})$ является ϕ-моделью, то мы будем называть её *наибольшей сингулярной ϕ-моделью*. LTS, множество ϕ-трасс которой равно наибольшей сингулярной ϕ-модели, будем называть *наибольшей сингулярной LTS*.
- *Сингулярной ϕ-моделью* будем называть ϕ-модель, вложенную во множество $\boldsymbol{d}\circ\boldsymbol{Sing}(\boldsymbol{\Sigma})$. LTS, множество ϕ-трасс которой является сингулярной ϕ-моделью, будем называть *сингулярной LTS*.

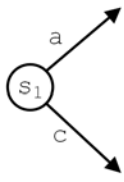
□

На Рис.70 приведены примеры сингулярных и несингулярных состояний, а также все соответствующие им предоминанты.

<table>
<tr><th colspan="3">Сингулярные и несингулярные состояния</th></tr>
<tr><td colspan="3">L={a,b,c,x,y,z}, $\mathfrak{R}$={{a},{c},{a,b},{b,c},{x,y,z}}</td></tr>
<tr><td>Несингулярное состояние</td><td colspan="2">Сингулярные состояния</td></tr>
<tr><td> 

Действия x и y несингулярны</td><td> 

Действие b несингулярно</td><td> 

Все действия сингулярны</td></tr>
<tr><td colspan="3">Предоминанты</td></tr>
<tr><td>{s},{s,s_1},{s,s_2},{s,s_1,s_2},{s_1,s_2}
 
</td><td>{s},{s,s_1}
 
</td><td>{s}</td></tr>
<tr><td colspan="3">Рис.70.</td></tr>
</table>

Множество сингулярных ϕ-трасс может быть не ***d***-замкнуто так же, как могут быть не ***d***-замкнуты множество однородных и множество финальных ϕ-трасс. К сожалению, в отличие от ***d***-замыканий $d\circ Final(\Sigma)$ и $d\circ Uniform(\Sigma)$, ***d***-замыкание $d\circ Sing(\Sigma)$ может не быть ϕ-моделью или мажорантной ϕ-моделью. Мы дадим несколько примеров, разъясняющих эту проблему. В то же время это множество, по определению, состоит из конформных ϕ-трасс.

Лемма 60: Множество $d\circ Sing(\Sigma)$ обладает всеми свойствами предельной ϕ-модели, кроме, быть может, конвергентности.

□556

Лемма 61: Если ϕ-трассы μ и $\mu`$ конформны и $\mu \preccurlyeq \mu`$, то из конформности ϕ-трассы $\mu`\cdot\lambda$ следует конформность ϕ-трассы $\mu\cdot\lambda$.

□557

Лемма 62: Если ***d***-замыкание множества сингулярных ϕ-трасс мажорантно, то оно является монотонной ϕ-моделью.

□558

Определим преобразование ϕ-модели, которое заменяет несингулярный ϕ-символ на его ближайшую предоминанту, каждый ϕ-символ которой получается добавлением в ***ref***-множество одного несингулярного действия.

Определение 120: Пусть ϕ-символ $\circ$ несингулярен. Обозначим через $np(\circ)$ множество ϕ-символов вида $\circ_z=(\circ_r\cup\{z\},\circ_g)$, где z пробегает множество всех несингулярных действий. Пусть во множестве ϕ-трасс $\mathbf{T}$ имеется ϕ-трасса μ, не завершающаяся ϕ-символом ($postf(\mu)=\epsilon$), но продолжаемая несингулярным ϕ-символом $\circ$ ($\mu\cdot\langle\circ\rangle\in\mathbf{T}$). Определим преобразование множества ϕ-трасс $\mathbf{T}$, получающееся заменой ϕ-символа $\circ$ на ϕ-символы из множества $np(\circ)$:

$Np(\mathbf{T},\mu,\circ) =_{def} (\mathbf{T}\setminus\mathbf{N})\cup\mathbf{S}$, где

$\mathbf{N}=\{\boldsymbol{\mu}\cdot\boldsymbol{\rho}\cdot\boldsymbol{\lambda}\in\mathbf{T}\mid \boldsymbol{pref}(\boldsymbol{\lambda})=\epsilon \ \&\ \boldsymbol{\rho}\in\circ^{*} \ \&\ \boldsymbol{\rho}\neq\epsilon\}$,

$\mathbf{S}=\{\boldsymbol{\mu}\cdot\boldsymbol{\rho}_z\cdot\boldsymbol{\lambda}\mid \boldsymbol{\mu}\cdot\boldsymbol{\rho}\cdot\boldsymbol{\lambda}\in\mathbf{N} \ \&\ \circ_z\in\boldsymbol{np}(\circ) \ \&\ \boldsymbol{\rho}_z\in\circ_z^{*} \ \&\ |\boldsymbol{\rho}_z|=|\boldsymbol{\rho}|\}$.

□

Лемма 63: Пусть задана ϕ-модель $\mathbf{T}$, в которой имеется ϕ-трасса $\boldsymbol{\mu}$, не завершающаяся ϕ-символом, но продолжаемая несингулярным ϕ-символом $\circ$. Тогда множество $\boldsymbol{Np}(\mathbf{T},\boldsymbol{\mu},\circ)$ также является ϕ-моделью, $\mathbf{T}\sim_{\mathfrak{R}}\boldsymbol{Np}(\mathbf{T},\boldsymbol{\mu},\circ)$ и $\mathbf{T}\preccurlyeq\boldsymbol{Np}(\mathbf{T},\boldsymbol{\mu},\circ)$.

□559

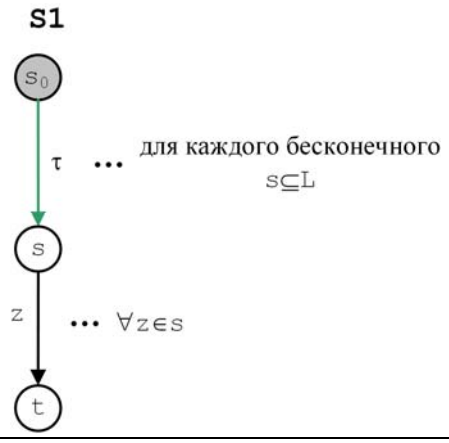
Как следствие, для любой монотонной ϕ-модели $\mathbf{T}$ множество $\boldsymbol{Np}(\mathbf{T},\boldsymbol{\mu},\circ)$ является монотонной ϕ-моделью, мажорирующей $\mathbf{T}$. Построение искомой сингулярной ϕ-модели сводится к множественному и многократному применению операции ***Np***. Проблема в том, что этот процесс может не закончиться, если каждый раз в ближайшей предоминанте несингулярного ϕ-символа снова оказывается несингулярный ϕ-символ.

Соотношение минимальных и сингулярных ϕ-моделей

В общем, сингулярные и минимальные мажорантные конформные ϕ-модели оказываются не связанными друг с другом. Это демонстрируют четыре приведённых ниже примера.

Пример 1: не существует минимальной (тем более, наименьшей) мажорантной конформной ϕ-модели, но ***d***-замыкание множества сингулярных ϕ-трасс является (наибольшей сингулярной) ϕ-моделью и она монотонна (Рис.71).

Соотношение минимальных и сингулярных ϕ-моделей. Пример 1



Алфавит L произвольное бесконечное множество. $\mathfrak{R}=\mathcal{P}(L)$.

Рис.71.

В этом примере из начального состояния s_0 проведён τ-переход в каждое состояние s, где s пробегает все бесконечные подмножества алфавита, а из каждого такого состояния s проведён переход в терминальное состояние t по каждому действию $z \in s$.

Нетрудно видеть, что спецификация $\mathbf{S1}$ содержит все конформные ϕ-трассы, и все они сингулярны: $\mathbf{\Phi}^{\omega}(\mathbf{S1}) = \boldsymbol{Conf} \circ \mathbf{\Phi}^{\omega}(\mathbf{S1}) = \boldsymbol{Sing} \circ \mathbf{\Phi}^{\omega}(\mathbf{S1})$.

Покажем, что не существует минимальной мажорантной подмодели. Выберем произвольное состояние s и произвольное действие $z \in s$. Рассмотрим ϕ-трассу $\boldsymbol{\mu} = \langle \phi(s), z \rangle$. Любая мажорантная подмодель должна содержать мажоранту $\boldsymbol{\mu}_1 \succcurlyeq \boldsymbol{\mu}$, которая, очевидно, имеет вид $\boldsymbol{\mu}_1 = \langle \phi(s_1), z \rangle$. Если в мажорантной подмодели есть строгая мажоранта $\boldsymbol{\mu}_2 \succ \boldsymbol{\mu}_1$, то мы можем удалить состояние s_1 (из подмодели удаляются все ϕ-трассы, порождаемые этим состоянием и только им), оставляя подмодель монотонной. Следовательно, такая подмодель не минимальна. Пусть теперь в мажорантной подмодели нет строгой мажоранты ϕ-трассы $\boldsymbol{\mu}_1$. Тогда, поскольку подмножество s_1 бесконечно, всегда найдётся его бесконечное собственное подмножество $s_2 \subset s_1$, содержащее z. Очевидно,

$\phi(s_1)=(L\backslash s_1,\varnothing)\prec(L\backslash s_2,\varnothing)=\phi(s_2)$ и $\boldsymbol{\mu}_2=\langle\phi(s_2),z\rangle\succ\boldsymbol{\mu}_1$. Если бы ϕ-трасса $\boldsymbol{\mu}_2$ имела в подмодели мажоранту $\boldsymbol{\mu}_3\succcurlyeq\boldsymbol{\mu}_2$, то, по транзитивности ϕ-мажорирования, было бы $\boldsymbol{\mu}_3\succ\boldsymbol{\mu}_1$, что не верно. Следовательно, ϕ-трасса $\boldsymbol{\mu}_2$ не мажорируется ϕ-трассами подмодели, что противоречит мажорантности подмодели.

Пример 2: существует минимальная мажорантная ϕ-модель, но ***d***-замыкание множества сингулярных ϕ-трасс не является ϕ-моделью (Рис.72).

В этом примере в LTS-спецификации **S2** из начального состояния s_0 проведён τ-переход в каждое состояние $X_n=\{x_1,\ldots x_n\}$, где $n=0,1,\ldots$, а из каждого такого состояния X_n проведён переход в терминальное состояние по каждому действию x_i, где $i>n$, а также переход по действию y в состояние T_n. Из последнего состояния ведут переходы в терминальное состояние по каждому действию $t\in T\backslash T_n$.

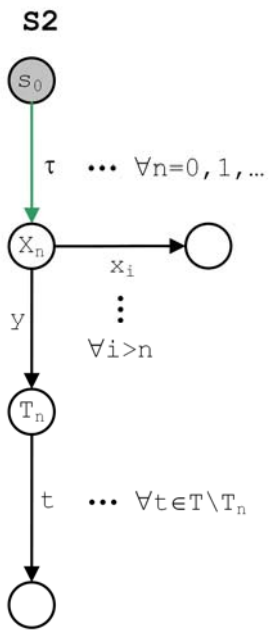
Только бесконечные подмножества множества T являются $\mathfrak{R}$-отказами, а во всех состояниях T_n только конечные подмножества множества T входят в ***ref***-множества. Поэтому, кроме ϕ-трасс спецификации $\boldsymbol{\Phi}^{\omega}(\mathbf{S2})$ конформными являются ϕ-трассы, получаемые замыканием по взятию префикса и ***dr***-операциями от ϕ-трасс вида $\boldsymbol{\mu}(n,k,t)=\langle(T\cup X_n,\varnothing),y,(T_k\cup X\cup\{y\},\varnothing),t,(L,\varnothing)\rangle$, где n и k независимо пробегают ряд чисел $0,1,\ldots$, а t пробегает $T\backslash T_k$.

Спецификация содержит только те ϕ-трассы $\boldsymbol{\mu}(n,k,t)$, в которых $n=k$.[32]

---

[32] Нам было бы достаточно выбрать для спецификации любое, но только одно, k для каждого n.

Соотношение минимальных и сингулярных ϕ-моделей. Пример 2



$L = X \cup \{y\} \cup T$,
где X и T произвольные счётные множества.

$X=\{x_1, x_2, \ldots\}$, $X_n=\{x_1, \ldots, x_n\}$, где $n=0,1,\ldots$ ($X_0=\varnothing$).

T_n – n-ое конечное подмножество множества T (множество конечных подмножеств счётного множества счётно), где $n=0,1,\ldots$.

$\mathfrak{R} = \{\{x_i\} | i=1,2,\ldots\} \cup \{\{y\}\} \cup \{U \subseteq T | |U|=\infty\}$.

Рис.72.

Спецификация $\boldsymbol{\Phi}^{\omega}(\mathbf{S2})$ мажорирует все конформные ϕ-трассы, то есть она является монотонной ϕ-моделью.

Действительно, любая конформная ϕ-трасса $\boldsymbol{\mu}(n,k,t)$ мажорируется ϕ-трассой спецификации $\boldsymbol{\mu}(m,m,t)$, где $m \geq n$ и $t \in T \setminus T_m$. Такое m всегда можно подобрать, поскольку для любого конечного подмножества T_k бесконечного множества T и любого $t \in T \setminus T_k$ существует бесконечное множество конечных подмножеств $T_m \subset T$, не содержащих t. Следовательно, одно из них имеет индекс $m \geq n$.

Более того, спецификация $\boldsymbol{\Phi}^{\omega}(\mathbf{S2})$ является минимальной мажорантной и конформной ϕ-моделью.

Поскольку для каждого i множество $\{x_i\}$ является $\mathfrak{R}$-отказом, любая ϕ-трасса спецификации вида $\langle (T \cup X_n, \varnothing), x_{n+1} \rangle$ не мажорируется никакой конформной ϕ-трассой, отличной от неё самой. Действительно, конформная мажоранта должна была бы иметь вид $\langle (T \cup X_m, \varnothing), x_{n+1} \rangle$, где $m \geq n$. Но такая ϕ-трасса конформна только, если $m=n$. Поэтому ϕ-трасса

$\langle(T\cup X_n,\varnothing),x_{n+1}\rangle$ должна быть в мажорантной конформной ϕ-модели. А тогда должна быть ϕ-трасса $\langle(T\cup X_n,\varnothing),y\rangle$. Эта ϕ-трасса в спецификации продолжается единственным ϕ-символом $(T_n\cup X\cup\{y\},\varnothing)$. Поэтому такое продолжение должно быть в любой мажорантной подмодели спецификации. Дальнейшие конформные продолжения определены однозначно.

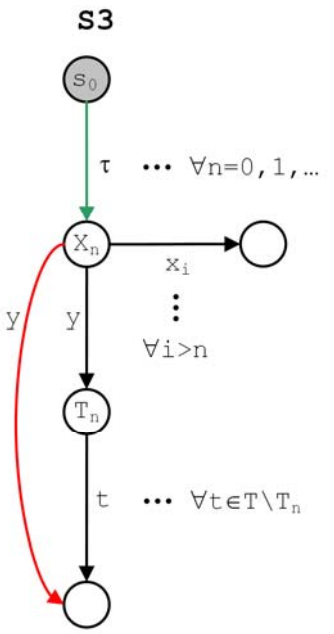
Поскольку только бесконечные подмножества множества T являются $\mathfrak{R}$-отказами, а во всех состояниях T_n только конечные подмножества множества T входят в ***ref***-множества, все их ϕ-символы $(T_n\cup X\cup\{y\},\varnothing)$ несингулярны. Поэтому сингулярные ϕ-трассы получаются замыканием по взятию префикса и ***dr***-операциями от ϕ-трасс вида $\langle(T\cup X_n,\varnothing),x_i,(L,\varnothing)\rangle$, где n пробегает ряд чисел $0,1,\ldots$, а $i>n$, и $\langle(T\cup X_n,\varnothing),y,t,(L,\varnothing)\rangle$, где n пробегает ряд чисел $0,1,\ldots$, а t пробегает T. Но тогда получается, что ϕ-трасса $\langle(T\cup X_n,\varnothing),y\rangle$ не продолжается сингулярным ϕ-символом (а также дивергенцией и разрушением). Поэтому множество $\boldsymbol{d}\circ\boldsymbol{Sing}(\boldsymbol{\Sigma})$ не конвергентно и, следовательно, не является ϕ-моделью.

Пример 3: существует минимальная мажорантная конформная ϕ-модель и ***d***-замыкание множества сингулярных ϕ-трасс является (наибольшей сингулярной) ϕ-модель, но она не мажорантна (Рис.73).

Отличие этого примера от предыдущего заключается в добавлении переходов $X_n\text{—}y\rightarrow$, ведущих в терминальное состояние (на рисунке слева выделены красным цветом). Исходная спецификация $\boldsymbol{\Phi}^{\omega}(\mathbf{S3})$, по-прежнему, является минимальной мажорантной конформной ϕ-моделью. Но теперь множество $\boldsymbol{d}\circ\boldsymbol{Sing}\circ\boldsymbol{\Phi}^{\omega}(\mathbf{S3})$ является ϕ-моделью, поскольку сингулярная ϕ-трасса $\langle(T\cup X_n,\varnothing),y\rangle$ продолжается терминальным сингулярным ϕ-символом $(L,\varnothing)$. Эта сингулярная модель изображена на Рис.73 справа.

Тем не менее, множество $d\circ Sing\circ\Phi^{\omega}(\mathbf{S3})$, по-прежнему, не мажорантно, так как ϕ-трассы вида $\langle(T\cup X_n,\varnothing),y,(T_n\cup X\cup\{y\},\varnothing),t\rangle$ не мажорируются им.

Соотношение минимальных и сингулярных ϕ-моделей. Пример 3



$L = X\cup\{y\}\cup T$,
где X и T произвольные счётные множества.

$X=\{x_1,x_2,\ldots\}$,
$X_n=\{x_1,\ldots,x_n\}$,
где $n=0,1,\ldots$ ($X_0=\varnothing$).

T_n – n-ое конечное подмножество множества T (множество конечных подмножеств счётного множества счётно),
где $n=0,1,\ldots$.

$\mathfrak{R} = \{\{x_i\}\,|\,i=1,2,\ldots\}\cup\{\{y\}\}\cup\{U\subseteq T\,||U|=\infty\}$.

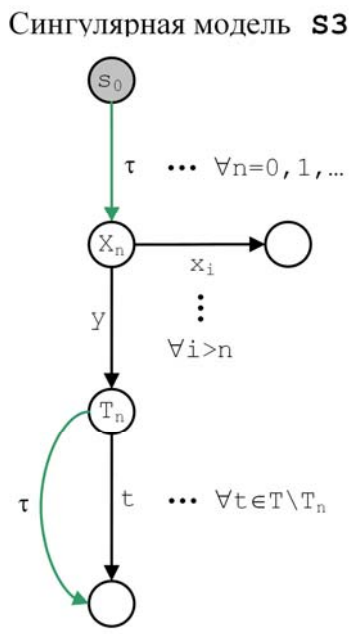


Рис.73.

Пример 4: существуют две минимальные мажорантные конформные ϕ-модели: сингулярная и несингулярная (Рис.74).

ϕ-модели $\mathbf{\Phi}^{\omega}(\mathbf{S4a})$ и $\mathbf{\Phi}^{\omega}(\mathbf{S4b})$ отличаются следующими ϕ-трассами (и их ***r***-замыканием):

S4a : $\boldsymbol{\mu}=\langle(\{x\},\varnothing),y,(\{a\},\varnothing),x\rangle$, $\boldsymbol{\lambda}=\langle(\{x\},\varnothing),y,(\{a\},\varnothing),y\rangle$;

S4b : $\boldsymbol{\sigma}=\langle(\{x\},\varnothing),y,(\{a,y\},\varnothing),x\rangle$.

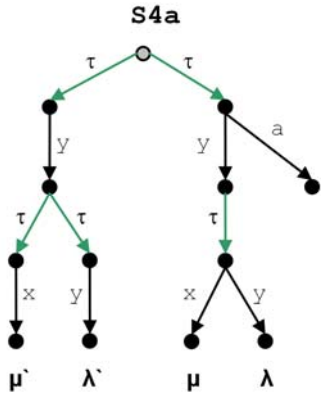
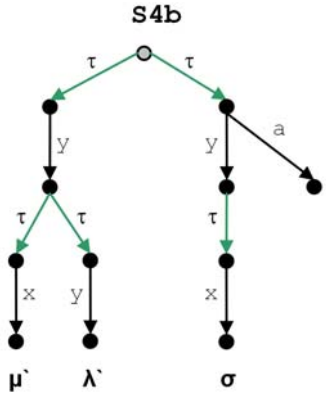
Соотношение минимальных и сингулярных ϕ-моделей. Пример 4	
$\mathbf{\Phi}^{\omega}(\mathbf{S4a})$ несингулярна	$\mathbf{\Phi}^{\omega}(\mathbf{S4b})$ сингулярна
	
$L=\{a,x,y\}$, $\mathfrak{R}=\{\{a\},\{x,y\}\}$	

Рис.74.

Однако, эти ϕ-трассы мажорируются общими ϕ-трассами (отличия выделены жирным шрифтом):

$\boldsymbol{\mu} \preccurlyeq \boldsymbol{\mu}` = \langle (\{\mathbf{a,}x\},\varnothing),y,(\{a\mathbf{,y}\},\varnothing),x\rangle$,

$\boldsymbol{\lambda} \preccurlyeq \boldsymbol{\lambda}` = \langle (\{\mathbf{a,}x\},\varnothing),y,(\{a\mathbf{,y}\},\varnothing),y\rangle$,

$\boldsymbol{\sigma} \preccurlyeq \boldsymbol{\mu}`$.

Поэтому эти ϕ-модели мажорируют друг друга. Как следствие, они конформны друг другу и могут рассматриваться как монотонные ϕ-модели для одной и той же спецификации.

В то же время обе они минимальны, так как нельзя удалить никакое непустое подмножество ϕ-трасс без нарушения свойств ϕ-модели, конформности или мажорантности.

При этом ϕ-модель $\mathbf{\Phi}^{\omega}(\mathbf{S4a})$ содержит ϕ-трассы $\boldsymbol{\mu}$ и $\boldsymbol{\lambda}$, которые не сингулярны и не могут быть получены из её сингулярных ϕ-трасс ***d***-операцией (не сингулярен ϕ-символ $(\{a\},\varnothing)$). В то же время ϕ-модель $\mathbf{\Phi}^{\omega}(\mathbf{S4b})$ состоит только из сингулярных ϕ-трасс.

Соотношение наименьшей и сингулярной ϕ-моделей

В отличие от минимальных, наименьшая мажорантная конформная ϕ-модель тесно связана с сингулярной ϕ-моделью, что показывается следующими двумя теоремами.

Теорема 38: Если наименьшая мажорантная конформная ϕ-модель существует, то она сингулярна.

□560

Теорема 39: Если существует наименьшая мажорантная конформная ϕ-модель, то ***d***-замыкание множества сингулярных ϕ-трасс является (наибольшей сингулярной) монотонной ϕ-моделью.

□561

Наименьшая мажорантная конформная ϕ-модель может как совпадать с наибольшей сингулярной ϕ-моделью, так и быть строго вложенной в неё. Это показывается примерами на Рис.75.

Здесь изображены все конформные ϕ-трассы, и обе они сингулярны. После состояний, отмеченных красным цветом, имеются маршруты с одной и той же трассой длины `n` слева и с двумя трассами: трассой длины `n` и более короткой трассой длины `m<n` справа. Слева наименьшая мажорантная конформная ϕ-модель не содержит ϕ-трасс, отмеченных пунктиром, поскольку они мажорируются остальными ϕ-трассами. Справа, из-за разной длины маршрутов после красных состояний, наименьшая мажорантная конформная ϕ-модель совпадает с наибольшей сингулярной ϕ-моделью.

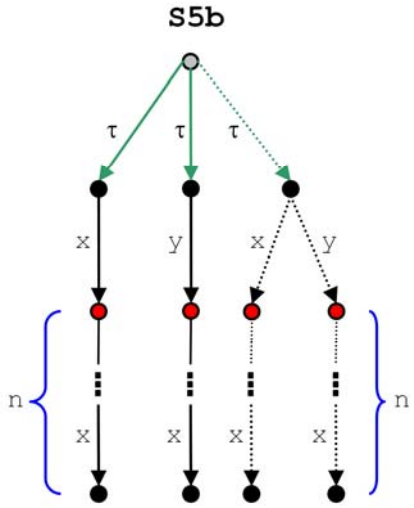
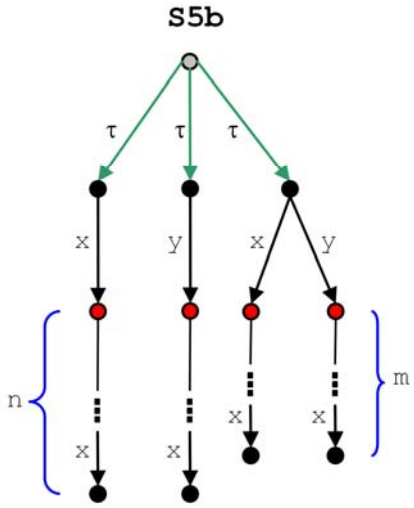
Соотношение наименьшей и наибольшей сингулярных ϕ-моделей. Пример 5	
Наименьшая мажорантная конформная модель строго вложена в наибольшую сингулярную	Наименьшая мажорантная конформная модель совпадает с наибольшей сингулярной
	
L={x,y}, $\mathfrak{R}$={{x},{y}}	

Рис.75.

Заметим, что различие левого и правого примеров обнаруживается после исследования маршрутов длины не менее m+2. Это означает, что при алгоритмическом монотонном преобразовании мы не сможем различить эти два случая на первых m+2 шагах, предполагая, что модель строится с постепенным увеличением длины трасс. Это является одной из причин, по которым мы в дальнейшем не будем стремиться строить наименьшую (или минимальную) мажорантную конформную модель, ограничиваясь наибольшей сингулярной моделью.

6.7. Спецификации с ограниченной ветвимостью

Структура раздела:

1. Конформно-конечно-ветвящиеся спецификации
2. LTS-модели
3. Минимальные ϕ-символы
4. Power-LTS
5. Доминанты и конечная сингуляризация

В этом разделе мы рассмотрим ограничения, которые можно наложить на спецификацию для того, чтобы ***d***-замыкание множества сингулярных ϕ-трасс было монотонной ϕ-моделью. Эти ограничения являются ограничениями на ветвимость модели. Здесь мы будем исходить из того, что при любом задании LTS с помощью локальных алгоритмов для алгоритмизации как генерации тестов, так и различных преобразований спецификации, нужна конечность числа состояний LTS-спецификации после каждой безопасной $\mathfrak{R}$-трассы. Вместе с тем заметим, что такое требование, вообще говоря, не является необходимым для существования наибольшей сингулярной монотонной ϕ-модели (что показывает пример 1 из предыдущего раздела).

Сначала мы сформулируем ограничения на наибольшую монотонную модель, а потом ограничения на исходную LTS-спецификацию. Во всём этом разделе, если не оговорено противное, будем предполагать, что заданы $\mathfrak{R}$-семантика и LTS-спецификация $\mathbf{S}$.

6.7.1. Конформно-конечно-ветвящиеся спецификации

Определение 121: Будем говорить, что спецификация *конформно-конечно-ветвящаяся* или *КК-ветвящаяся*, если в наибольшей однородной ϕ-модели каждая безопасная ϕ-трасса продолжается конечным числом ϕ-символов.

□

Лемма 64: Если спецификация КК-ветвящаяся, то ***d***-замыкание множества её сингулярных ϕ-трасс мажорантно.

□562

Теорема 40: Если спецификация КК-ветвящаяся, то ***d***-замыкание множества её сингулярных ϕ-трасс является (наибольшей сингулярной) монотонной ϕ-моделью.

□564

6.7.2. LTS-модели с ограниченной ветвимостью

Мы уже отмечали (4.7.4), что для алгоритмической генерации тестов требуется Б.БЛК+Б.О-ветвимость и ТБ.П-ветвимость LTS-спецификации. Сейчас мы сузим класс рассматриваемых спецификаций до Б.БЛК+Б.О-ветвящихся.

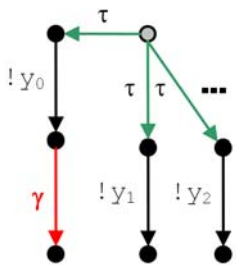
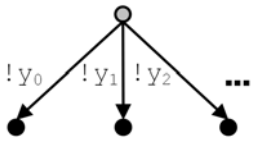
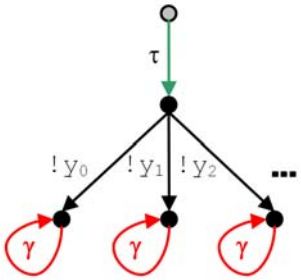
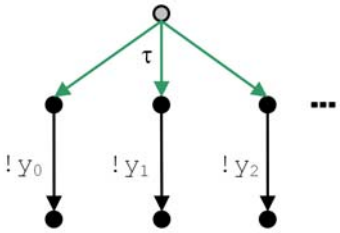
Теорема 41: Для того, чтобы спецификация была КК-ветвящейся необходимо и достаточно, чтобы каноническая наибольшая однородная LTS была Б.БЛК+Б.О-ветвящейся.

□564

В канонической наибольшей однородной LTS, например, $\Phi 2L \circ d \circ Uniform \circ \Phi^{\omega}(\mathbf{s})$, бесконечное ветвление возможно только как бесконечный «веер» τ-переходов из нестабильного состояния в стабильные состояния с разными ϕ-символами. Нам нужно, чтобы такой «веер» был конечным в каждом нестабильном состоянии, достижимом по безопасной ϕ-трассе. Достижимость по безопасной ϕ-трассе эквивалентна достижимости по безопасной $\mathfrak{R}$-трассе (Б-достижимости), причём безопасную $\mathfrak{R}$-трассу можно понимать как безопасную $\mathfrak{R}$-трассу спецификации **s** (Теорема 18:).

Будем считать, что LTS-спецификация **s** Б.БЛК+Б.О-ветвящаяся, поскольку это всё равно требуется для алгоритмической генерации тестов. Прежде всего, отметим, что это свойство спецификации (даже если она каноническая) само по себе не является ни достаточным, ни необходимым условием Б.БЛК+Б.О-ветвимости наибольшей однородной канонической LTS. Это условие не необходимо, поскольку требуется конечность числа однородных ϕ-символов, а не всех конформных ϕ-символов, что показывается примером спецификации **s1** в βγδ-семантике слева на Рис.76. Также это условие не достаточно, поскольку сингуляризация даже одного ϕ-символа может привести к бесконечному множеству сингулярных ϕ-символов, что

показывается примером спецификации **S2** в той же βγδ-семантике справа на Рис.76.

<table>
<tr><td colspan="2">Для Б.БЛК+Б.О-ветвимости наибольшей однородной спецификации
условие Б.БЛК+Б.О-ветвимости исходной спецификации</td></tr>
<tr><td>не необходимо</td><td>не достаточно</td></tr>
<tr><td>S1
 
</td><td>S2
 
</td></tr>
<tr><td>Каноническая LTS для
d∘Uniform∘Φ^ω(S1)
 
</td><td>Каноническая LTS для
d∘Uniform∘Φ^ω(S2)
 
</td></tr>
<tr><td colspan="2">L={!y1,!y2,…}, 𝔑={δ}={L}</td></tr>
<tr><td colspan="2">Рис.76.</td></tr>
</table>

Далее мы будем исследовать те ограничения, которые нужно наложить на Б.БЛК+Б.О-ветвящуюся LTS-спецификацию **S**, чтобы каноническая наибольшая однородная LTS также была Б.БЛК+Б.О-ветвящейся.

6.7.3. Минимальные ϕ-символы

Исследуем ϕ-символы, которыми может продолжаться безопасная 𝔑-трасса в наибольшей однородной модели. Формально продолжение безопасной 𝔑-трассы ϕ-символом определяется следующим образом.

Определение 122: Будем говорить, что безопасная $\mathfrak{R}$-трасса μ продолжается ϕ-символом o, если в некоторой конформной ϕ-модели $\mathbf{\Sigma}$ $\mathfrak{R}$-трасса μ генерируется некоторой ϕ-трассой $\boldsymbol{\mu}\in\mathbf{\Sigma}$, то есть $\mu\in\xi(\boldsymbol{\mu})$, которая продолжается ϕ-символом o, то есть $\boldsymbol{\mu}\cdot\langle o\rangle\in\mathbf{\Sigma}$.

□

Очевидно, $\mathfrak{R}$-трасса продолжается ϕ-символом тогда и только тогда, когда в некоторой конформной LTS она заканчивается в состоянии с этим ϕ-символом.

Безопасную $\mathfrak{R}$-трассу μ, не завершающуюся $\mathfrak{R}$-отказом ($\boldsymbol{postf}(\mu)=\epsilon$), будем называть тау-трассой. Рассмотрим такую тау-трассу и определим те ϕ-символы, которыми она может продолжаться в наибольшей однородной модели. Сначала мы определим минимальные по ***ref***-множеству однородные ϕ-символы, из которых все остальные однородные ϕ-символы получаются сингуляризацией, то есть входят во множество ϕ-символов, доминирующих минимальные ϕ-символы (их верхние конусы по отношению доминирования). Мы будем различать два случая в зависимости от того, порождает или не порождает минимальный ϕ-символ хотя бы один $\mathfrak{R}$-отказ.

Сначала дадим ряд вспомогательных определений.

Определение 123: Пусть $\mu\in\boldsymbol{Safe}_{\circ}\boldsymbol{F}(\mathbf{S})$ безопасная $\mathfrak{R}$-трасса.

- Обозначим множество внешних действий, опасных после $\mathfrak{R}$-трассы μ:

  $\boldsymbol{gamma}(\mu) =_{\text{def}} \{z\in L \mid z\ \boldsymbol{\sout{safe}}\ \boldsymbol{F}(\mathbf{S})\ \boldsymbol{after}\ \mu\}$.
- Обозначим множество безопасных внешних действий, которыми не продолжается $\mathfrak{R}$-трасса μ:

  $\boldsymbol{ref}(\mu) =_{\text{def}} \{z\in L \mid \mu\cdot\langle z\rangle\notin\boldsymbol{F}(\mathbf{S})\}\setminus\boldsymbol{gamma}(\mu)$.
- Условие конформности продолжения $\mathfrak{R}$-трассы μ дивергенцией:

$\boldsymbol{div}(\mu) =_{\text{def}} \mu\cdot\langle\Delta\rangle\in \boldsymbol{F}(\mathbf{S}) \vee \boldsymbol{gamma}(\mu)=\mathbb{L} \;\&\; \varnothing\notin\mathfrak{R}$.

- $\mathfrak{R}$-трассу μ будем называть *тау-трассой*, если она не завершается $\mathfrak{R}$-отказом. Множество тау-трасс обозначим:
  $\boldsymbol{Tau}(\mathbf{S}) =_{\text{def}} \{\mu\in \boldsymbol{Safe}\circ\boldsymbol{F}(\mathbf{S}) \mid \boldsymbol{postf}(\mu)=\epsilon\}$.
- Тау-трассу μ будем называть *полной трассой*, если она продолжается хотя бы одним действием из каждого $\mathfrak{R}$-отказа:
  μ полна $=_{\text{def}} \mu\in \boldsymbol{Tay}(\mathbf{S}) \;\&\; \mathfrak{R}\circ\boldsymbol{ref}(\mu)=\varnothing$.
- Безопасную $\mathfrak{R}$-трассу μ будем называть *стабильной трассой*, если она завершается $\mathfrak{R}$-отказом (не тау-трасса) или полна. Множество стабильных трасс обозначим:
  $\boldsymbol{Stab}(\mathbf{S}) =_{\text{def}} \{\mu\in \boldsymbol{Safe}\circ\boldsymbol{F}(\mathbf{S}) \mid \boldsymbol{postf}(\mu)\neq\epsilon \vee \mu \text{ полна}\}$.

□

Если есть пустая кнопка $\varnothing\in\mathfrak{R}$, полных трасс не бывает, так как трасса не может продолжаться действием из пустого $\mathfrak{R}$-отказа. В этом случае стабильные трассы – это те безопасные трассы, которые завершаются $\mathfrak{R}$-отказом, быть может, пустым.

Лемма 65: Для $\mathfrak{R}$-трассы μ условие $\boldsymbol{div}(\mu)=\boldsymbol{true}$ необходимо, если μ безопасна, и всегда достаточно для конформности продолжения $\mathfrak{R}$-трассы μ дивергенцией.

□566

Определение 124: Пусть $\mu\in \boldsymbol{Tau}(\mathbf{S})$, а $\rho\in\mathfrak{R}^*$ трасса $\mathfrak{R}$-отказов такая, что $\mu\cdot\rho\in \boldsymbol{Stab}(\mathbf{S})$. Определим для тау-трассы μ *минимальный ϕ-символ*, порождающий трассу ρ:

$\phi(\mu\cdot\rho) =_{\text{def}} (\boldsymbol{ref}(\mu\cdot\rho), \boldsymbol{gamma}(\mu\cdot\rho))$.

Для тау-трассы μ *доминирующими ϕ-символами* будем называть ϕ-символы, доминирующие её минимальные ϕ-символы $\phi(\mu\cdot\rho)$, то есть элементы верхних конусов минимальных ϕ-символов $\phi(\mu\cdot\rho)^{\triangle}$.

□

Для полной тау-трассы μ ***ref***-множество её минимального ϕ-символа $\phi(\mu)$ не порождает $\mathfrak{R}$-отказов: $\mathfrak{R}\circ\phi(\mu)=\varnothing$. В остальных случаях, когда $\rho\neq\epsilon$, ***ref***-множество минимального ϕ-символа $\phi(\mu\cdot\rho)$ порождает, по крайней мере, $\mathfrak{R}$-отказы из непустого множества $\boldsymbol{Im}(\rho)$.

Определение 125: Пусть тау-трасса μ продолжается ϕ-символом $\circ$. Трассу $\mathfrak{R}$-отказов ρ будем называть $(\mu,\circ)$-безопасной, если $\rho\in\xi(\circ)$ и $\mathfrak{R}$-трасса $\mu\cdot\rho$ безопасна. $(\mu,\circ)$-безопасную трассу ρ будем называть *главной трассой ϕ-символа* $\circ$, если для каждой $(\mu,\circ)$-безопасной трассы $\rho`$ имеет место $\mathbf{S}$ ***after*** $\mu\cdot\rho\subseteq\mathbf{S}$ ***after*** $\mu\cdot\rho`$. Множество главных трасс ϕ-символа $\circ$ будем обозначать $\zeta(\mu,\circ)$.

□

Лемма 66: Пусть спецификация $\mathbf{S}$ Б.БЛК+Б.О-ветвящаяся. Тогда для любой тау-трассы μ и любого продолжающего её ϕ-символа $\circ$, существует главная трасса ϕ-символа $\rho\in\zeta(\mu,\circ)$.

□567

Определение 126: Безопасную $\mathfrak{R}$-трассу μ, генерируемую конечной безопасной ϕ-трассой $\boldsymbol{\mu}$, будем называть главной трассой ϕ-трассы $\boldsymbol{\mu}$, если для каждой безопасной $\mathfrak{R}$-трассы $\mu`$, генерируемой $\phi$-трассой $\boldsymbol{\mu}$, имеет место $\mathbf{S}$ ***after*** $\mu\subseteq\mathbf{S}$ ***after*** $\mu`$. Множество главных трасс ϕ-трассы $\boldsymbol{\mu}$ будем обозначать $\zeta(\boldsymbol{\mu})$.

□

Главную трассу ϕ-трассы можно также определить индуктивно:

a) если ϕ-трасса продолжается действием, то её главная трасса также продолжается этим действием

$\mu\in\zeta(\boldsymbol{\mu}) \Leftrightarrow \mu\cdot\langle z\rangle\in\zeta(\boldsymbol{\mu}\cdot\langle z\rangle)$,

b) если ϕ-трасса не завершается ϕ-символом, но продолжается ϕ-символом, то её главная трасса продолжается главной трассой этого ϕ-символа

$\boldsymbol{postf}(\boldsymbol{\mu})=\epsilon \;:\; \mu\in\zeta(\boldsymbol{\mu}) \;\&\; \rho\in\zeta(\mu,o) \Leftrightarrow \mu\cdot\rho\in\zeta(\boldsymbol{\mu}\cdot\langle o\rangle)$,

c) если ϕ-трасса завершается ϕ-символом и продолжается (тем же) ϕ-символом, её главная трасса не меняется $\zeta(\boldsymbol{\mu}\cdot\langle o,o\rangle)=\zeta(\boldsymbol{\mu}\cdot\langle o\rangle)$.

Лемма 67: Пусть спецификация $\boldsymbol{s}$ Б.БЛК+Б.О-ветвящаяся. Тогда для любой конечной безопасной ϕ-трассы $\boldsymbol{\mu}$ существует её главная трасса $\mu\in\zeta(\boldsymbol{\mu})$.

□568

Лемма 68: Пусть спецификация $\boldsymbol{s}$ Б.БЛК+Б.О-ветвящаяся. Внешнее действие безопасно после конечной безопасной ϕ-трассы $\boldsymbol{\mu}$ тогда и только тогда, когда оно безопасно после её главной трассы $\mu\in\zeta(\boldsymbol{\mu})$.

□569

Лемма 69: Пусть спецификация $\boldsymbol{s}$ Б.БЛК+Б.О-ветвящаяся. Если конечная однородная ϕ-трасса $\boldsymbol{\mu}\cdot\langle o\rangle$ завершается ϕ-символом o, то $o_g=\boldsymbol{gamma}(\mu)$, где $\mu\in\zeta(\boldsymbol{\mu}\cdot\langle o\rangle)$ главная трасса.

□570

Следующая Лемма показывает, что в наибольшей однородной модели каждый ϕ-символ доминирует некоторый минимальный ϕ-символ, но, правда, не после любой продолжаемой им тау-трассы, а только после главной трассы ϕ-трассы.

Лемма 70: Пусть спецификация $\boldsymbol{s}$ Б.БЛК+Б.О-ветвящаяся. Тогда для любой конечной однородной ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$, завершающейся ϕ-символом o, этот ϕ-

символ доминирует минимальный ϕ-символ главной трассы ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$: для $\mu\in\zeta(\boldsymbol{\mu}\cdot\langle o\rangle)$ имеет место $\phi(\mu)\trianglelefteq o$.

□570

Лемма 71: Если конечная однородная ϕ-трасса $\boldsymbol{\mu}\cdot\langle a\rangle$ завершается ϕ-символом a, а ϕ-символ b доминирует его, $a\trianglelefteq b$, то ϕ-трасса $\boldsymbol{\mu}\cdot\langle b\rangle$ также однородна.

□571

Теперь мы построим такую однородную модель, в которой имеется каждая тау-трасса $\mu\in\boldsymbol{Tau}(\mathbf{s})$, она является главной трассой некоторой ϕ-трассы и продолжается каждым своим минимальным ϕ-символом. Отсюда будет следовать, что, если LTS-спецификация Б.БЛК+Б.О-ветвящаяся, то в наибольшей однородной модели каждая тау-трасса продолжается теми и только теми ϕ-символами, которые входят в верхние конусы по доминированию минимальных ϕ-символов. Заметим, что без ограничения на ветвимость спецификации в наибольшей однородной модели тау-трасса μ может продолжаться недоминирующими ϕ-символами. Пример изображён на Рис.77.

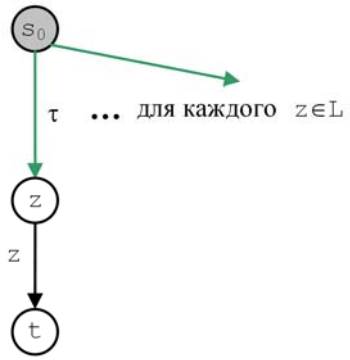
Без ограничений на ветвимость тау-трасса может продолжаться недоминирующим ϕ-символом	
	L – произвольное бесконечное множество действий. $\mathfrak{R}$ = {{z} | z∈L}. Переходы: $s_0 \text{—}\tau\text{→} z$, где z пробегает L; $z \text{—} z\text{→} t$, где z пробегает L. Все ϕ-символы вида (***ref***, ∅), где ***ref***⊆L, конформны. ϕ-символ минимален, если ***ref*** конечно. Минимальный ϕ-символ доминируется только им самим. Все ϕ-символы с бесконечным ***ref*** не доминирующие.

Рис.77.

6.7.4. Power-LTS

Идея построения искомой модели заключается в следующем. Для LTS-спецификации $\mathbf{S}$ мы будем строить power-LTS $\boldsymbol{P}(\mathbf{S})$. Её состояния мы будем называть power-состояниями. Они соответствуют множествам состояний после тау-трасс $\boldsymbol{Tay}(\mathbf{S})$ и стабильных трасс $\boldsymbol{Stab}(\mathbf{S})$. Кроме того, добавим ещё одно специальное состояние γ, которое будет соответствовать продолжениям безопасных трасс опасными действиями. Из power-состояния, соответствующего множеству состояний $\mathbf{S}\ \boldsymbol{after}\ \sigma$, где σ безопасна, проводим переход по действию z, если a) хотя бы в одном состоянии $s\in(\mathbf{S}\ \boldsymbol{after}\ \sigma)$ есть переход $s\text{—}z\rightarrow s`$, или b) действие $z$ опасно после $\sigma$.[33] Концом этого перехода будет в случае a) power-состояние, соответствующее множеству таких постсостояний $s`$, то есть множеству $\mathbf{S}\ \boldsymbol{after}\ \sigma\cdot\langle z\rangle$, а в случае b) состояние γ, в котором проведём γ-петлю. Нестабильное power-состояние будет соответствовать тау-трассе, а стабильное состояние – стабильной трассе. Для этого из power-состояния, соответствующего множеству состояний $\mathbf{S}\ \boldsymbol{after}\ \mu$, где μ тау-трасса, проводим τ-переход в power-состояние, соответствующее множеству состояний $\mathbf{S}\ \boldsymbol{after}\ \mu\cdot\rho$, где ρ трасса отказов и $\mu\cdot\rho$ стабильная трасса. Поскольку полная трасса μ является одновременно тау-трассой и стабильной трассой ($\rho=\epsilon$), нам нужно как-то различать соответствующие ей нестабильное и стабильное power-состояния. Нестабильное power-состояние для тау-трассы μ мы будем обозначать как $\mathbf{S}\ \boldsymbol{after}\ \mu$, а стабильное power-состояние для стабильной трассы $\mu\cdot\rho$ – как $(\mathbf{S}\ \boldsymbol{after}\ \mu\cdot\rho, \boldsymbol{ref}(\mu\cdot\rho))$. Если тау-трасса μ конформно продолжается дивергенцией, то в power-состоянии $\mathbf{S}\ \boldsymbol{after}\ \mu$ проведём τ-

---

[33] То есть для каждого $\mathfrak{R}$-отказа P, содержащего z, в одном из состояний $s\in(\mathbf{S}\ \boldsymbol{after}\ \sigma)$ есть переход $s\text{—}z`\rightarrow s`=\langle\gamma\rangle\Rightarrow$, где $z`\in P$.

петлю. Начальным power-состоянием объявим power-состояние **S** ***after*** ϵ, если в LTS **S** нет трассы $\langle\gamma\rangle$, или состояние γ в противном случае.

Замечание. Построение power-LTS аналогично построению power-графа, которое иногда называют «детерминизацией». Эта техника используется для построения детерминированного графа (или соответствующего автомата Мили), порождающего то же множество последовательностей, что исходный граф (автомат) [52,113]. Если единственный вид наблюдений – это внешние действия, то есть 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 из нестабильного состояния в стабильные состояния с разными ϕ-символами (наличие даже одного достижимого τ-перехода делает LTS недетерминированной).

Далее мы покажем, что любое стабильное power-состояние (**S** ***after*** $\mu\cdot\rho$, ***ref***$(\mu\cdot\rho)$) имеет минимальный ϕ-символ $\phi(\mu\cdot\rho)$. Также покажем, что в power-состоянии **S** ***after*** μ заканчивается ϕ-трасса с главной трассой μ.

Определение 127: Для сокращения записи множество состояний LTS $\mathbf{s}$ после $\mathfrak{R}$-трассы μ, когда LTS $\mathbf{s}$ подразумевается, будем кратко обозначать $[\mu]=(\mathbf{s}\ \textit{after}\ \mu)$.

□

Определение 128: Для LTS $\mathbf{s}=\mathrm{LTS}(V_{\mathbf{s}},L_\gamma,E_{\mathbf{s}},s_0)$ определим power-LTS $\boldsymbol{P}(\mathbf{s})=\mathbf{P}=\mathrm{LTS}(V_{\mathbf{P}},L_\gamma,E_{\mathbf{P}},p_0)$, где $V_{\mathbf{P}}=\mathcal{P}(V_{\mathbf{s}})\cup(\mathcal{P}(V_{\mathbf{s}})\times\mathcal{P}(L))\cup\{\gamma\}$, $p_0=[\epsilon]$, если $\langle\gamma\rangle\notin\boldsymbol{F}(\mathbf{s})$, или $p_0=\gamma$, если $\langle\gamma\rangle\in\boldsymbol{F}(\mathbf{s})$, а $E_{\mathbf{P}}$ – наименьшее множество, порождаемое следующими правилами вывода:

$\forall\mu\in\textit{Tau}(\mathbf{s})\quad\forall z\in L\quad\forall\rho\in\mathfrak{R}^*\quad\forall r\subseteq L$

(1) $\vdash\ \gamma\text{—}\gamma\!\rightarrow\!\gamma;$

(2) $\textit{div}(\mu)\quad\vdash\ [\mu]\text{—}\tau\!\rightarrow\![\mu];$

(3) $\mu\cdot\rho\in\textit{Stab}(\mathbf{s})\quad\vdash\ [\mu]\text{—}\tau\!\rightarrow\!([\mu\cdot\rho],\textit{ref}(\mu\cdot\rho));$

(4) $z\notin\textit{ref}(\mu)\ \ \&\ z\notin\textit{gamma}(\mu)\quad\vdash\ [\mu]\text{—}z\!\rightarrow\![\mu\cdot\langle z\rangle];$

(5) $z\notin\textit{ref}(\mu\cdot\rho)\ \ \&\ z\notin\textit{gamma}(\mu\cdot\rho)\quad\vdash\ ([\mu\cdot\rho],r)\text{—}z\!\rightarrow\![\mu\cdot\rho\cdot\langle z\rangle];$

(6) $z\in\textit{gamma}(\mu)\quad\vdash\ [\mu]\text{—}z\!\rightarrow\!\gamma;$

(7) $z\in\textit{gamma}(\mu\cdot\rho)\quad\vdash\ ([\mu\cdot\rho],r)\text{—}z\!\rightarrow\!\gamma.$

□

Отметим, что в power-состоянии $([\mu\cdot\rho],r)$ всегда $r=\textit{ref}(\mu\cdot\rho)$. В дальнейшем, когда мы будем выполнять сингуляризацию, это уже будет не обязательно так.

Теперь нам нужно показать: 1) power-LTS конформна, 2) power-LTS (финальна и) однородна, 3) каждая тау-трасса $\mu\in\textit{Tau}(\mathbf{s})$ имеется в power-LTS, заканчивается в состоянии $[\mu]$ и является главной трассой некоторой ϕ-трассы, 4) для каждого её стабильного продолжения трассой отказов

$\mu\cdot\rho\in\boldsymbol{Stab}(\mathbf{S})$ имеется стабильное power-состояние $([\mu\cdot\rho],\boldsymbol{ref}(\mu\cdot\rho))$ с минимальным ϕ-символом $\phi(\mu\cdot\rho)$.

Лемма 72: В power-LTS $\boldsymbol{P}(\mathbf{S})$ power-состояние вида $[\mu]$ или γ нестабильно, а power-состояние вида $([\mu],r)$ стабильно.

□572

Лемма 73: Пусть в power-LTS $\boldsymbol{P}(\mathbf{S})$ конечный маршрут R имеет нормальную ϕ-трассу $\boldsymbol{\mu}$ и заканчивается в power-состоянии $[\mu]$ или $([\mu],r)$. Тогда $\mu\in\xi(\boldsymbol{\mu})$, $\mu\in\boldsymbol{Safe}\circ\boldsymbol{F}(\mathbf{S})$ и для каждой $\mathfrak{R}$-трассы $\sigma\in\xi(\boldsymbol{\mu})$ имеет место $\sigma\in\boldsymbol{F}(\mathbf{S})$ & $[\mu]\subseteq[\sigma]$.

□573

Лемма 74: $\boldsymbol{P}(\mathbf{S})$ ***saco*** $\mathbf{S}$.

□575

Лемма 75: $\boldsymbol{P}(\mathbf{S})$ однородна.

□576

Лемма 76: Каждая тау-трасса $\mu\in\boldsymbol{Tau}(\mathbf{S})$ имеется в power-LTS $\boldsymbol{P}(\mathbf{S})$, заканчивается в power-состоянии $[\mu]$ и является главной трассой некоторой ϕ-трассы.

□578

Лемма 77: Для каждой тау-трассы $\mu\in\boldsymbol{Tau}(\mathbf{S})$ и каждого её стабильного продолжения трассой отказов $\mu\cdot\rho\in\boldsymbol{Stab}(\mathbf{S})$ в power-LTS $\boldsymbol{P}(\mathbf{S})$ имеется стабильное power-состояние $([\mu\cdot\rho],\boldsymbol{ref}(\mu\cdot\rho))$, в котором заканчивается трасса $\mu\cdot\rho$ и которое имеет минимальный ϕ-символ $\phi(\mu\cdot\rho)$.

□579

Теорема 42: Для Б.БЛК+Б.О-ветвящейся LTS-спецификации каждая тау-трасса продолжается в наибольшей однородной модели всеми своими доминирующими ϕ-символами, то есть ϕ-символами из верхних конусов по доминированию минимальных ϕ-символов, и только такими ϕ-символами.

□579

6.7.5. Доминанты и конечная сингуляризация

Далее нам нужно ответить на два вопроса: 1) когда число минимальных ϕ-символов после тау-трассы конечно, 2) когда в верхнем конусе по доминированию минимального ϕ-символа можно найти минимальное конечное подмножество, доминирующее все ϕ-символы этого конуса. Такое подмножество будем называть *минимальной конечной доминантой*.[34] Тогда мы можем заменить минимальный ϕ-символ на его доминанту.

Определение 129: Для ϕ-символа a его *доминантой* будем называть такое подмножество $A \subseteq a^{\triangle}$ его верхнего конуса по доминированию, которое включает предоминанты всех ϕ-символов этого верхнего конуса:

$\forall b \in a^{\triangle}\ \exists B \subseteq A$ B предоминанта b,

то есть $B \subseteq b^{\triangle}\ \&\ \forall z \in L \setminus (b_r \cup b_g)\ \exists c \in B\ z \notin c_r$.

□

Формальное условие из этого определения эквивалентно следующему условию:

$A \subseteq a^{\triangle}\ \&\ \forall b \in a^{\triangle}\ (\exists c \in A\ b \trianglelefteq c)\ \&\ (\forall z \in L \setminus b_r\ \exists c \in A\ b \trianglelefteq c\ \&\ z \in L \setminus c_r)$.

Прежде всего, отметим, что, если доминанты бесконечны, то может не существовать сингулярной доминанты. Например, пусть алфавит L бесконечен, а семейство $\mathfrak{R}$ – это все бесконечные подмножества алфавита. Тогда рассмотрим ϕ-символ $o=(a,\varnothing)$ с конечным ***ref***-множеством a и пустым ***gamma***-множеством. Этот ϕ-символ o имеет бесконечное множество несингулярных действий. Любое конечное подмножество b несингулярных действий можно добавить в ***ref***-множество a и получить доминирующий ϕ-

[34] Напомним, что предоминанта ϕ-символа доминирует его, но не обязательно доминирует все ϕ-символы из его верхнего конуса.

символ. Однако этот ϕ-символ $(a \cup b, \varnothing)$ будет несингулярным (у него, по-прежнему, бесконечное число несингулярных действий). В то же время добавление бесконечного множества b несингулярных действий к ***ref***-множеству a меняет множество порождаемых им $\mathfrak{R}$-отказов и, тем самым, не даёт доминирующего ϕ-символа. Таким образом, в верхнем конусе ϕ-символа o вообще нет сингулярных ϕ-символов.

По соображениям алгоритмизации нам нужно, чтобы искомая монотонная модель была Б.БЛК+Б.О-ветвящейся (например, для того, чтобы по ней также можно было генерировать тесты). Поэтому мы не будем исследовать вопрос о том, при каких условиях существует сингулярная доминанта, если она может быть бесконечной. Нас будут интересовать только конечные доминанты.

Для ответа на первый вопрос (когда число минимальных ϕ-символов после тау-трассы конечно) отметим, что, по Лемме 76 и Лемме 77, в power-LTS каждая тау-трасса μ продолжается всеми своими минимальными ϕ-символами как ϕ-символами power-состояний вида $([\mu \cdot \rho], \mathit{ref}(\mu \cdot \rho))$, расположенными на концах «веера» τ-переходов из power-состояния $[\mu]$. Поэтому для ответа на первый вопрос достаточно следующей Леммы.

Лемма 78: Для Б.БЛК+Б.О-ветвящейся LTS-спецификации power-LTS ЛК+О-ветвящаяся.

□580

Очевидно, что ЛК+О-ветвимость влечёт Б.БЛК+Б.О+ТБ.П-ветвимость. На самом деле, дерево τ-маршрутов, начинающихся в достижимом состоянии такой LTS, не только перечислимо-ветвящееся, но даже конечно-ветвящееся и содержит конечное число состояний.

Если мы найдём ответ на второй вопрос (когда минимальный ϕ-символ имеет минимальную конечную доминанту), мы сможем наложить такие дополнительные ограничения на LTS-спецификацию, что будет возможно дальнейшее преобразование «расщепления» состояния с минимальным ϕ-символом на конечное множество состояний с ϕ-символами из его конечной

минимальной доминанты. Ниже мы покажем, что, если такая конечная доминанта существует, то весь верхний конус оказывается конечным, минимальная (очевидно, конечная) доминанта является наименьшей и совпадает с подмножеством верхнего конуса, состоящим из всех сингулярных доминирующих ϕ-символов.

Лемма 79: Если число несингулярных действий ϕ-символа бесконечно, то любая доминанта этого ϕ-символа бесконечна.

□581

Лемма 80: Необходимым и достаточным условием конечности верхнего конуса ϕ-символа по доминированию является конечность множества несингулярных действий этого ϕ-символа.

□582

Лемма 81: Если существует конечная доминанта ϕ-символа a, то множество сингулярных ϕ-символов из $a^{\triangle}$ является наименьшей доминантой.

□582

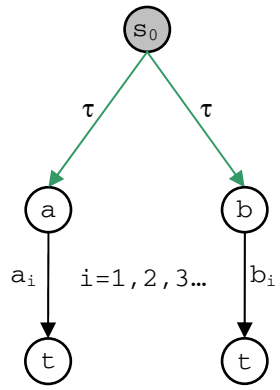
Определение 130: Будем говорить, что LTS-спецификация *конечно-доминируема*, если она Б.БЛК+Б.О-ветвящаяся и для каждой тау-трассы каждый минимальный ϕ-символ имеет конечное число несингулярных действий.

□

Таким образом, искомое ограничение на LTS-спецификацию $\mathbf{S}$ сводится к её конечно-доминируемости.

Заметим, что в спецификации $\mathbf{S}$ тау-трасса может продолжаться не только минимальными ϕ-символами и может не продолжаться какими-то минимальными ϕ-символами. Более того, тау-трасса может продолжаться в исходной спецификации $\mathbf{S}$ только такими ϕ-символами, которые имеют конечное число несингулярных действий, однако спецификация не будет конечно-доминируемой. Это показывает пример на Рис.78.

Все ϕ-символы исходной LTS-спецификации сингулярны,
но спецификация не конечно-доминируема



L=A∪B, где $A=\{a_1,a_2,\ldots\}$ и $B=\{b_1,b_2,\ldots\}$ счётные множества.
$\mathfrak{R} = \{\{a_i,b_i\}\,|\,i=1,2,\ldots\}$.
$\phi(a)=(B,\varnothing)$ и $\phi(b)=(A,\varnothing)$ сингулярны.
Пустая трасса ϵ полна.
$\phi(\epsilon)=(\varnothing,\varnothing)$ минимален,
но имеет бесконечное число несингулярных действий.

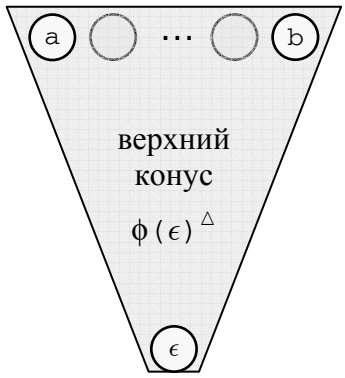


Рис.78.

Теорема 43: Если LTS-спецификация конечно-доминируема, то существует монотонная ЛК+О-ветвящаяся (и, следовательно, Б.БЛК+Б.О+ТБ.П-ветвящаяся) наибольшая сингулярная LTS-модель.

□584

На самом деле, для Б.БЛК+Б.О-ветвящихся LTS-спецификаций конечно-доминируемость необходимо для существования любой монотонной Б.БЛК+Б.О-ветвящейся однородной LTS-модели, а не только наибольшей сингулярной.

Лемма 82: Пусть спецификация **s** Б.БЛК+Б.О-ветвящаяся. В наибольшей однородной модели мажорирование ϕ-трасс эквивалентно равенству ***gamma***-множеств и вложенности ***ref***-множеств соответствующих ϕ-символов:

$$\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma} \Leftrightarrow (|\boldsymbol{\mu}|=|\boldsymbol{\sigma}| \;\&\; \forall i\in[1..|\boldsymbol{\mu}|]\ \boldsymbol{\mu}(i)=\boldsymbol{\sigma}(i)\in L \vee$$

$$\boldsymbol{\mu}(i)\in\Phi(L) \;\&\; \boldsymbol{\sigma}(i)\in\Phi(L) \;\&\; \boldsymbol{\mu}(i)_r\subseteq\boldsymbol{\sigma}(i)_r \;\&\; \boldsymbol{\mu}(i)_g=\boldsymbol{\sigma}(i)_g).$$

□585

Лемма 83: Пусть имеется множество `B` (не обязательно попарно различных) элементов `b(i,j)`, где индексы `i` и `j` независимо пробегают бесконечное множество индексов `I`, причём `i≠j`. Если для каждых `i≠j` и `i≠k` выполнено `b(i,j)≠b(k,i)`, то множество `B` бесконечно.

□587

Теорема 44: Если спецификация Б.БЛК+Б.О-ветвящаяся, но не конечно-доминируема, то не существует монотонной Б.БЛК+Б.О-ветвящейся однородной LTS-модели.

□588

Теорема 45: Если существует монотонная Б.БЛК+Б.О-ветвящаяся LTS-модель, то существует такая же однородная LTS-модель.

□589

6.8. Монотонное преобразование

Структура раздела:

1. Определение монотонного преобразования
2. Оптимизация
3. Преобразование при многократной композиции
4. Алгоритмизация

В этом разделе, подводя итоги, определим формально преобразование исходной LTS-спецификации, и те ограничения на исходную спецификацию, при выполнении которых результат преобразования даёт наибольшую сингулярную LTS-спецификацию. Также рассмотрим коротко вопросы оптимизации преобразования и его алгоритмизации.

6.8.1. Определение монотонного преобразования

Определение 131: Для ϕ-символа o обозначим множество доминирующих его сингулярных ϕ-символов:

sing$(o) =_{def} \{s \in \Phi(L) \mid o \trianglelefteq s \ \&\ s$ сингулярен$\}$.

□

Определение 132: Для LTS $\mathbf{S}=LTS(V_{\mathbf{S}}, L_\gamma, E_{\mathbf{S}}, s_0)$ определим LTS $\mathcal{T}(\mathbf{S})=\mathbf{T}=LTS(V_{\mathbf{T}}, L_\gamma, E_{\mathbf{T}}, t_0)$, где $V_{\mathbf{T}}=\mathcal{P}(V_{\mathbf{S}}) \cup (\mathcal{P}(V_{\mathbf{S}}) \times \mathcal{P}(L)) \cup \{\gamma\}$, $t_0=[\epsilon]$, если $\langle\gamma\rangle \notin \boldsymbol{F}(\mathbf{S})$, или $t_0=\gamma$, если $\langle\gamma\rangle \in \boldsymbol{F}(\mathbf{S})$, а $E_{\mathbf{T}}$ – наименьшее множество, порождаемое следующими правилами вывода:

$\forall \mu \in$ ***Tau***$(\mathbf{S})$ $\forall z \in L$ $\forall \rho \in \mathfrak{R}^*$ $\forall s \in \Phi(L)$

(1) $\vdash \gamma \text{—}\gamma\text{→}\gamma$;

(2) ***div***(μ) $\vdash [\mu] \text{—}\tau\text{→} [\mu]$;

(3) $\mu\cdot\rho \in$ ***Stab***$(\mathbf{S})$ & $s \in$ ***sing***$\circ$***ref***$(\mu\cdot\rho)$ $\vdash [\mu] \text{—}\tau\text{→} ([\mu\cdot\rho], s_r)$;

(4) $z \notin$ ***ref***(μ) & $z \notin$ ***gamma***(μ) $\vdash [\mu] \text{—}z\text{→} [\mu\cdot\langle z\rangle]$;

(5) $z \notin s_r$ & $z \notin$ ***gamma***$(\mu\cdot\rho)$ $\vdash ([\mu\cdot\rho], s_r) \text{—}z\text{→} [\mu\cdot\rho\cdot\langle z\rangle]$;

(6) $z \in$ ***gamma***(μ) $\vdash [\mu] \text{—}z\text{→}\gamma$;

(7) $z \in$ ***gamma***$(\mu\cdot\rho)$ $\vdash ([\mu\cdot\rho], s_r) \text{—}z\text{→}\gamma$.

□

Состояния преобразованной LTS $\mathcal{T}(\mathbf{S})$ мы будем называть power-состояниями, так же как для power-LTS.

Мы налагаем на LTS-спецификацию условие конечно-доминируемости:

1) спецификация Б.БЛК+Б.О-ветвящаяся,
2) для каждой тау-трассы μ каждый её минимальный ϕ-символ имеет конечное число несингулярных действий.

Множество несингулярных действий ϕ-символа зависит только от него самого и семейства $\mathfrak{R}$.

Минимальные ϕ-символы вычисляются как ϕ-символы power-состояний, соответствующих множествам состояний вида $\mathbf{S}$ ***after*** $\mu\cdot\rho$, где $\mathfrak{R}$-трасса $\mu\cdot\rho$ стабильна, то есть безопасна и либо полна, либо завершается $\mathfrak{R}$-отказом.

Лемма 84: Каждое power-состояние преобразованной LTS $\mathcal{T}(\mathbf{S})$, кроме состояния γ, Б-достижимо.

□590

Теорема 46: Для конечно-доминируемой LTS-спецификации $\mathbf{S}$ преобразованная LTS $\mathcal{T}(\mathbf{S})$ монотонная, предельная, ЛК+О-ветвящаяся.

□590

Посмотрим, какое монотонное преобразование получается для примеров, которые рассматривались в начале главы.

Для примера (Рис.62) составной системы из передатчика $\mathbf{T}$ и приёмника $\mathbf{F}$ монотонное преобразование выполняет сингуляризацию. В начальном состоянии исходной спецификации определены два действия из одного «кнопочного» множества: две реакции в передатчике $\mathbf{T}_0$ для семантики, в которой тест принимает все реакции, и два стимула в приёмнике $\mathbf{F}_0$ для семантики, в которой тест посылает два стимула. Преобразование добавляет два τ-перехода в новые состояния, в каждом из которых определяется только одно из этих двух действий. Получаются монотонные спецификации $\mathbf{T}_3$ и $\mathbf{F}_3$.

На Рис.61 был приведён пример несохранения конформности при асинхронном тестировании системы $\mathbf{S}$ через ограниченную выходную очередь

с «обнулением» **Q**. Здесь монотонное преобразование также, как в случае передатчика **T**, выполняет сингуляризацию реакций (Рис.79).

Монотонное преобразование для асинхронного тестирования через выходную очередь с «обнулением»

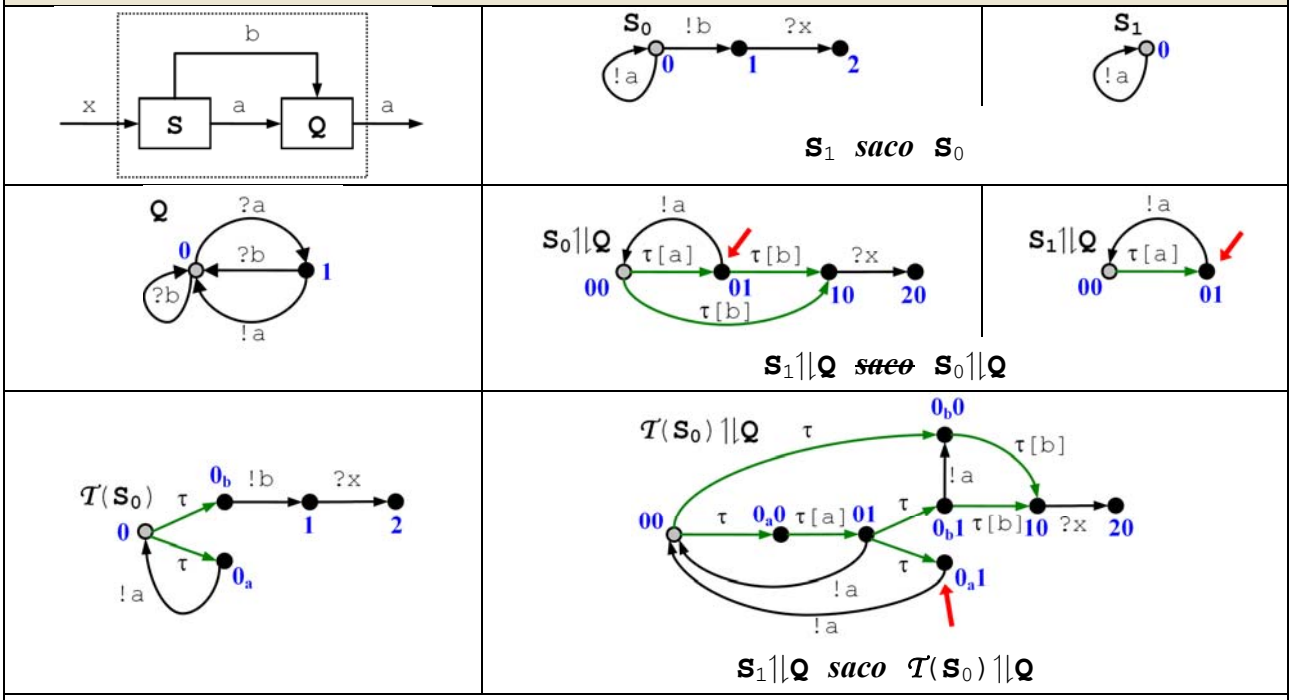


Рис.79.

6.8.2. Оптимизация

Прежде всего, заметим, что для получения наибольшей сингулярной LTS-спецификации можно вычислять не минимальные ϕ-символы, а ϕ-символы, соответствующие всем подмножествам стабильных состояний после тау-трассы и множеству всех состояний после полной тау-трассы. Это может привести лишь к тому, что некоторые состояния на конце «веера» τ-переходов окажутся заведомо эквивалентными по ϕ-трассам: множества ϕ-трасс, начинающихся в состояниях `s` и `s`, будут совпадать **Φ**(`s`)=**Φ**(`s`). Зато перебор всех подмножеств стабильных состояний конечного множества состояний легче поддаётся алгоритмизации, чем перебор всех минимальных ϕ-символов.

В то же время эквивалентные по ϕ-трассам состояния могут появиться даже при вычислении только минимальных ϕ-символов. LTS-модель без

эквивалентных состояний называется минимальной, для её построения требуется просмотр маршрутов вперёд на любое число шагов (переходов). Такую оптимизацию можно делать алгоритмически, очевидно, только для конечных LTS. При просмотре на конечное заранее фиксированное число шагов n мы можем выполнять частичную оптимизацию, которая в общем случае не гарантирует получение минимальной LTS. При такой частичной оптимизации два состояния s и $s`$ считаются n-эквивалентными, если выполнены следующие условия:

1) в них начинаются одни и те же ϕ-трассы с подтрассой базовых символов длины не более n:

   $\boldsymbol{\Phi}_n(s)=\boldsymbol{\Phi}_n(s`)$,

   где $\boldsymbol{\Phi}_n(t)=\{\boldsymbol{\mu}\in\boldsymbol{\Phi}(t)\mid|\boldsymbol{\mu}^{\downarrow}L_{\gamma\Delta}|\le n\}$;

2) ϕ-трассы с подтрассой базовых символов длины не более n одинаково продолжаются разрушением:

   $\boldsymbol{\Phi}_{n\gamma}(s)=\boldsymbol{\Phi}_{n\gamma}(s`)$, где $\boldsymbol{\Phi}_{n\gamma}(t)=\{\boldsymbol{\mu}\in\boldsymbol{\Phi}_{n+1}(t)\mid\boldsymbol{\mu}(|n+1|)=\gamma\}$;

3) ϕ-трассы с подтрассой базовых символов длины не более n, не продолжаемые разрушением, одинаково продолжаются дивергенцией:

   $\boldsymbol{\Phi}_{n\Delta}(s)=\boldsymbol{\Phi}_{n\Delta}(s`)$,

   где $\boldsymbol{\Phi}_{n\Delta}(t)=\{\boldsymbol{\mu}\in\boldsymbol{\Phi}_{n+1}(t)\setminus\boldsymbol{\Phi}_{n\gamma}(t)\mid\boldsymbol{\mu}(|n+1|)=\Delta\}$;

4) после каждой ϕ-трассы $\boldsymbol{\mu}$ с подтрассой базовых символов длины n, не продолжаемой разрушением и дивергенцией, одинаковые множества состояний с точностью до терминальных состояний:

   $s\ \boldsymbol{after}_n\ \boldsymbol{\mu}\ =\ s`\ \boldsymbol{after}_n\ \boldsymbol{\mu}$,

   где $t\ \boldsymbol{after}_n\ \boldsymbol{\mu}\ =\{u\in(t\ \boldsymbol{after}\ \boldsymbol{\mu})\mid\boldsymbol{init}(u)\neq\varnothing\}$.

Понятно, что, если состояния n-эквивалентны для некоторого n, то они полностью эквивалентны. Обратное, однако, в общем случае не верно (пример на Рис.80).

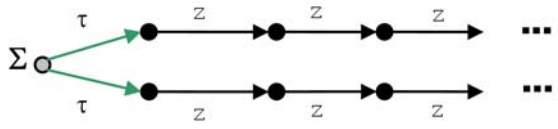
Эквивалентные, но не n-эквивалентные, состояния	
	Каждое состояние верхнего ряда эквивалентно находящемуся под ним состоянию нижнего ряда.

Рис.80.

Частным случаем такой оптимизации является удаление «лишних» нестабильных состояний, из которых не выходят переходы по внешним действиям и выходит только один τ-переход в стабильное состояние.

Ещё одной очевидной оптимизацией является удаление «лишних» переходов $s \xrightarrow{z}$ по действию z из нестабильного power-состояния s, если переходы по действию z ведут только из стабильных состояний, соответствующих этому power-состоянию s. В этом случае достаточно остающихся переходов по этому действию из стабильных power-состояний, расположенных на концах «веера» τ-переходов $s \xrightarrow{\tau}$.

В общем, все эти возможные оптимизации не принципиальны и относятся в большей степени к алгоритмизации монотонного преобразования.

6.8.3. Преобразование при многократной композиции

Композиция преобразованных LTS, вообще говоря, может не быть преобразованной LTS (её нельзя получить преобразованием $\mathcal{T}$ из какой-нибудь LTS). Достаточно рассмотреть свойство сингулярности состояний. В примере на Рис.81 (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.

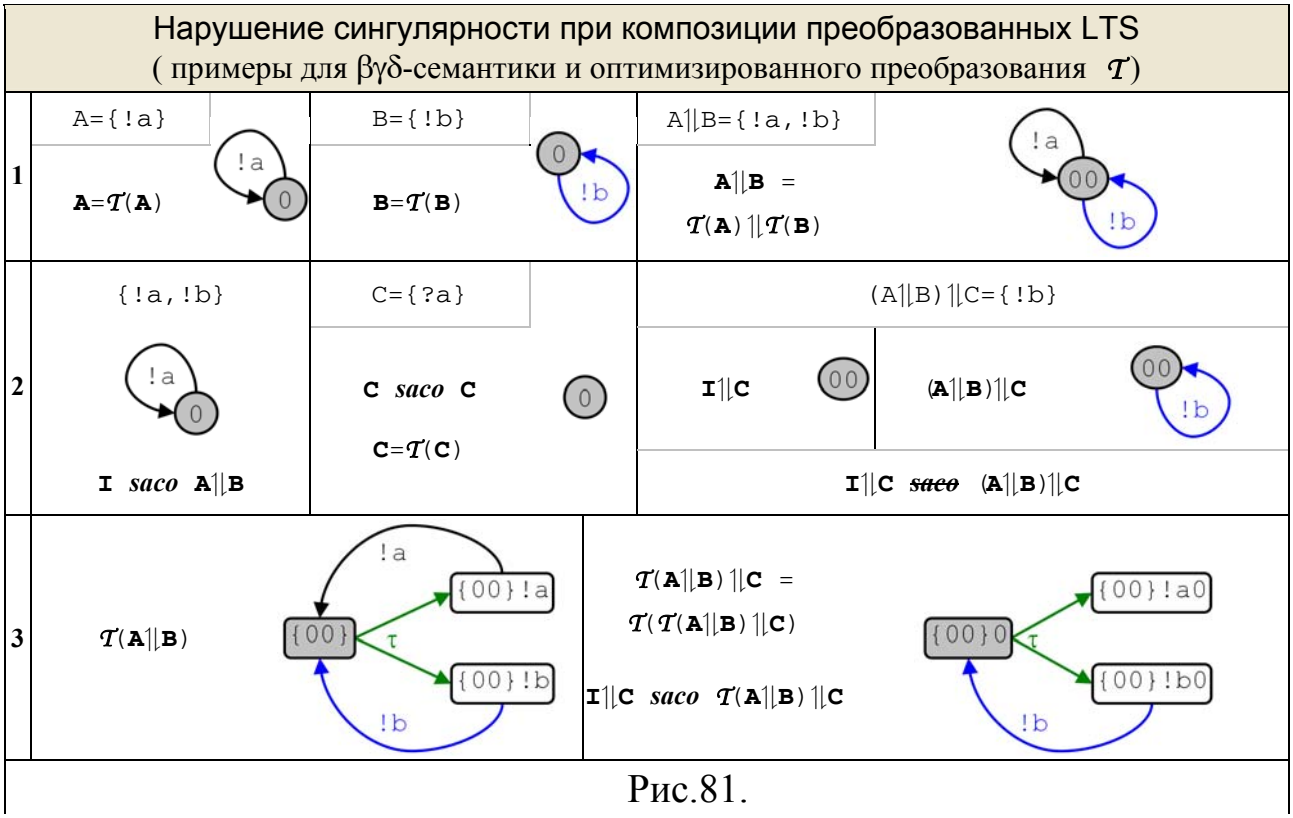


Рис.81.

Что меняется при преобразовании композиции преобразованных спецификаций, то есть чем отличаются $\mathcal{T}(\mathbf{S_1})||\mathcal{T}(\mathbf{S_2})$ и $\mathcal{T}(\mathcal{T}(\mathbf{S_1})||\mathcal{T}(\mathbf{S_2}))$?

Мы доказали, что ϕ-трассы композиции $\mathbf{S_{12}}=\mathcal{T}(\mathbf{S_1})||\mathcal{T}(\mathbf{S_2})$ конформны и мажорируют все ϕ-трассы всех композиций конформных реализаций, то есть ϕ-трассы композиций $\mathbf{I_1}||\mathbf{I_2}$, где $\mathbf{I_1}$ ***saco*** $\mathbf{S_1}$ и $\mathbf{I_2}$ ***saco*** $\mathbf{S_2}$. 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 $\mathcal{T}(\mathbf{S_3})$, если мы рассматриваем только те реализации, конформные $\mathbf{S_{12}}$, которые могут быть представлены как композиция $\mathbf{I_1}||\mathbf{I_2}$ реализаций, конформных спецификациям $\mathbf{S_1}$ и $\mathbf{S_2}$.

Иными словами, если схема компоновки, то есть число компонентов n и порядок их композиции, считаются фиксированными, 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 компоновки. Для $\mathbf{I_1}$ *saco* $\mathbf{S_1}$, $\mathbf{I_2}$ *saco* $\mathbf{S_2}$, ..., $\mathbf{I_n}$ *saco* $\mathbf{S_n}$ всегда будет $\mathbf{I_1}\|\mathbf{I_2}\|...\|\mathbf{I_n}$ *saco* $\mathcal{T}(\mathbf{S_1})\|\mathcal{T}(\mathbf{S_2})\|...\|\mathcal{T}(\mathbf{S_n})$, где опущенные скобки, определяющие порядок композиции[35], должны быть расставлены одинаково для реализаций и спецификаций в соответствии с данной схемой компоновки.

Иными словами, если композиционная система состоит из подсистем, которые, в свою очередь, композиционны, то для того, чтобы не было повторных преобразований, мы должны знать не только спецификации подсистем, полученные композицией преобразованных спецификаций их компонентов, но и схему компоновки каждой подсистемы из её компонентов. В этом случае преобразуются только спецификации «атомарных» компонентов самого нижнего уровня композиционной системы (не являющиеся подсистемами, то есть не разложимые далее на компоненты).

Однако, если схема компоновки реализаций не обязательно совпадает со схемой компоновки спецификаций, могут быть и другие реализации, конформные спецификации системы. Вообще, для спецификации $\mathbf{S_{12}}$ могут существовать конформные ей 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 $\mathcal{T}$-преобразования «промежуточных» результатов предшествующих композиций может привести к нарушению монотонности. Пример на Рис.81 (строка **2**). Поэтому для дальнейшей композиции нам требуется преобразовать композицию, то есть рассматривать спецификацию $\mathcal{T}(\mathbf{S_{12}})$ (на Рис.81 строка **3**).

Аналогичный пример можно привести для нарушения свойства однородности (Рис.82).

---

[35] Скобки нужны, поскольку композиция LTS, вообще говоря, не ассоциативна.

Нарушение однородности при композиции преобразованных LTS
(примеры для βγδ-семантики и оптимизированного преобразования $\mathcal{T}$)

A={!a}	B={!b}	A⫴B={!a,!b}
A=$\mathcal{T}$(**A**)	**B**=$\mathcal{T}$(**B**)	**A⫴B** = $\mathcal{T}$(**A**)⫴$\mathcal{T}$(**B**)

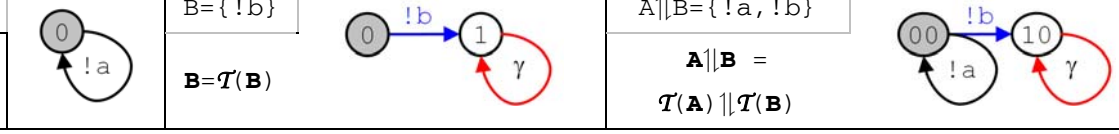


Рис.82.

Иными словами, если нас интересуют только спецификации подсистем, полученные композицией преобразованных спецификаций их компонентов, но не интересует внутреннее устройство этих подсистем, то есть мы допускаем иные схемы компоновки этих подсистем и неконформные реализации компонентов (при условии конформности самой подсистемы), то для получения спецификации системы нам требуется преобразовывать спецификации подсистем. В этом случае композиция спецификаций выполняется по схеме компоновки спецификаций с помощью оператора $.⌈\mathcal{T}⌋. = \mathcal{T}(.)⫴\mathcal{T}(.)$, то есть как $\mathbf{S_1}⌈\mathcal{T}⌋\mathbf{S_2}⌈\mathcal{T}⌋...⌈\mathcal{T}⌋\mathbf{S_n}$ (с требуемой расстановкой скобок).

6.8.4. Алгоритмизация

Предлагаемое в данной работе преобразование LTS-спецификации является монотонным для конечно-доминируемых (и, следовательно, Б.БЛК+Б.О+ТБ.П-ветвящихся) LTS. Результатом такого преобразования является ЛК+О-ветвящаяся LTS и, следовательно, Б.БЛК+Б.О+ТБ.П-ветвящаяся LTS, что требуется для алгоритмической генерации тестов по преобразованной спецификации. Здесь мы рассмотрим алгоритмизацию монотонного преобразования.

Б.БЛК+Б.О+ТБ.П-ветвимость LTS-спецификации гарантирует, что тау-трасса μ и стабильная трасса $\mu{\cdot}\rho$ заканчиваются в конечных множествах состояний. Рассмотрим, что нужно сделать для определения перехода по каждому внешнему действию z из power-состояния $[\mu]$ или power-

состояния $([\mu\cdot\rho], s_r)$ и для определения «веера» τ-переходов из нестабильного power-состояния $[\mu]$ во все стабильные power-состояния вида $[\mu\cdot\rho]$.

Переход по внешнему действию z.

Прежде всего, мы должны определить, является ли действие z опасным или безопасным после $\mathfrak{R}$-трассы μ или $\mu\cdot\rho$, соответственно. Для этого нужно каким-то образом решить следующую *задачу 1*: определить 1) каким $\mathfrak{R}$-отказам принадлежит действие z, 2) для каждого такого $\mathfrak{R}$-отказа P определить, какие ещё действия $z` \in P$, и 3) проверить, не достижим ли γ-переход $s` \text{—}\gamma\rightarrow$ по τ-переходам из постсостояния какого-нибудь перехода $s \text{—}z\rightarrow s`$ по внешнему действию $z`$ из какого-нибудь состояния $s \in [\mu]$ или $s \in [\mu\cdot\rho]$.

Для алгоритмического выполнения последнего пункта (для фиксированного действия $z`$) требуется ТБ.П-ветвимость LTS.

Если действие опасно, мы проводим переход $[\mu] \text{—}z\rightarrow\gamma$ или $([\mu\cdot\rho], s_r) \text{—}z\rightarrow\gamma$, соответственно. Если действие безопасно, мы определяем множество состояний $[\mu\cdot\langle z\rangle]$ или $[\mu\cdot\rho\cdot\langle z\rangle]$, соответственно, и проводим переходы $[\mu] \text{—}z\rightarrow [\mu\cdot\langle z\rangle]$ или $([\mu\cdot\rho], s_r) \text{—}z\rightarrow [\mu\cdot\rho\cdot\langle z\rangle]$, соответственно. Для этого нужен итератор, перечисляющий для заданного состояния s и заданного действия z все переходы вида $s \text{—}z\rightarrow s`$, то есть все постсостояния $s`$.

«Веер» τ-переходов $[\mu] \text{—}\tau\rightarrow ([\mu\cdot\rho], s_r)$.

Конечно-доминируемость LTS-спецификации гарантирует, что этот «веер» τ-переходов конечен. Формально, для получения всех таких τ-

переходов нужно перечислить все трассы $\mathfrak{R}$-отказов ρ, безопасно продолжающих тау-трассу μ. Даже если мы ограничимся трассами без повторных отказов, таких трасс ρ может быть бесконечное число (за счёт ***i***-операции). Вместе с тем число подмножеств состояний $[\mu\cdot\rho]$ конечно. Более того, конечно вообще семейство подмножеств стабильных состояний множества $[\mu]$. Проблема лишь в том, что не всякое такое подмножество является подмножеством состояний после какой-нибудь безопасной трассы $\mu\cdot\rho$.

Мы могли бы сравнивать эти подмножества $A_1, A_2, \ldots$, определяя ϕ-символы соответствующих power-состояний $[A_1], [A_2], \ldots$. Такое сравнение всегда можно выполнить алгоритмически для конечного алфавита, но не всегда – для бесконечного алфавита. Мы можем избежать этого сравнения, если, как было указано в предыдущем подразделе, будем использовать все подмножества стабильных состояний за счёт введения заведомо эквивалентных power-состояний. Для выбранного подмножества состояний мы должны решить следующую *задачу 2*: 1) определить соответствующий ϕ-символ и 2) провести его сингуляризацию. Кроме этого мы должны решить *задачу 3*: определить, не является ли тау-трасса μ полной.

Если алфавит конечен, все три указанные задачи можно выполнить алгоритмически за конечное время. Для бесконечного алфавита требуется задание специальных алгоритмов и/или предположение об «устройстве» семейства $\mathfrak{R}$ и/или LTS-спецификации.

Например, для βγδ-семантики достаточно конечности числа реакций в алфавите при бесконечном числе стимулов [27]. Можно разрешить и бесконечное число реакций в алфавите, если конечно число переходов по реакциям в каждом Б-достижимом состоянии LTS-спецификации и задан алгоритм, перечисляющий эти переходы в каждом таком состоянии.

В более общем случае можно ограничиться Б.К+Б.О+ТБ.П-ветвящимися LTS-спецификациями, то есть потребовать конечности множества всех переходов, определённых в Б-достижимом состоянии. Б.К+Б.О+ТБ.П-ветвящуюся LTS можно определять с помощью трёх алгоритмов: 1) алгоритма, перечисляющего конечное множество переходов для каждого Б-достижимого состояния, 2) алгоритма, перечисляющего перечислимое множество τ-переходов, определённых в ТБ-достижимом состоянии, и 3) алгоритма, указывающего наличие или отсутствие γ-перехода из ТБ-достижимого состояния. Тем самым, можно перечислить за конечное время множество внешних действий, по которым есть переходы из конечного и замкнутого по τ-переходам множества состояний после безопасной $\mathfrak{R}$-трассы или некоторого его подмножества стабильных состояний (число таких подмножеств конечно). Далее, опираясь на ТБ.П-ветвимость LTS, можно разбить это множество на два непересекающихся подмножества внешних действий: разрушающие (хотя бы в одном состоянии) и неразрушающие (в каждом состоянии) действия. Тогда для решения указанных трёх задач требуются три алгоритма, работающие только с такой парой конечных непересекающихся множеств внешних действий и семейством $\mathfrak{R}$.

Вопросы алгоритмизации при бесконечном алфавите довольно сложны и более детальное их рассмотрение выходит за рамки темы данной работы. Можно лишь подчеркнуть, что не существует практически приемлемого общего решения для любой $\mathfrak{R}$-семантики и любой конечно-доминируемой LTS-спецификации. В то же время для того или иного сочетания ограничений на $\mathfrak{R}$-семантику и ограничений на LTS-спецификацию часто можно предложить практически приемлемые и эффективные алгоритмы задания семантики и спецификации, на базе которых строится алгоритм монотонного преобразования.

6.9. Композиция

Структура раздела:

1. Ветвимость результата композиции
2. Ограничения на преобразованные LTS
3. Ограничения на исходные LTS
4. Алгоритмизация
5. Проблема сохранения безопасности при композиции

Композиция преобразованных спецификаций, помимо прочего, может использоваться для генерации системных тестов. В связи с этим возникают две проблемы.

1. Для алгоритмической генерации тестов по композиции монотонно преобразованных спецификаций требуется Б.БЛК+Б.О+ТБ.П-ветвимость результата композиции. Однако без дополнительных ограничений на спецификации компонентов это требование может быть нарушено.
2. Системное тестирование по композиции монотонно преобразованных спецификаций должно быть безопасным. Если мы предполагаем не конформность компонентов, а только их безопасность, то нужно, чтобы безопасность реализации системы относительно косой композиции спецификаций компонентов следовала из безопасности реализаций компонентов относительно спецификаций компонентов. К сожалению, это не всегда так: система гарантированно безопасна только при конформных, а не просто безопасных, реализациях компонентов. Заметим, что, если компоненты конформны, то системное тестирование излишне, так как композиция таких компонентов, как доказано выше, гарантированно конформна косой композиции спецификаций.

   Данный раздел посвящён исследованию этих двух проблем.

6.9.1. Ветвимость результата композиции

Для алгоритмической генерации тестов требуется Б.БЛК+Б.О+ТБ.П-ветвимость композиции. На Рис.83 приведены примеры, когда композиция Б.БЛК+Б.О+ТБ.П-ветвящихся LTS не обладает этим свойством. Эта проблема имеет два аспекта:

1) появление бесконечной цепочки τ-переходов, проходящей через бесконечное число различных состояний (на рисунке вверху),
2) появление бесконечного «веера» τ-переходов (на рисунке внизу).

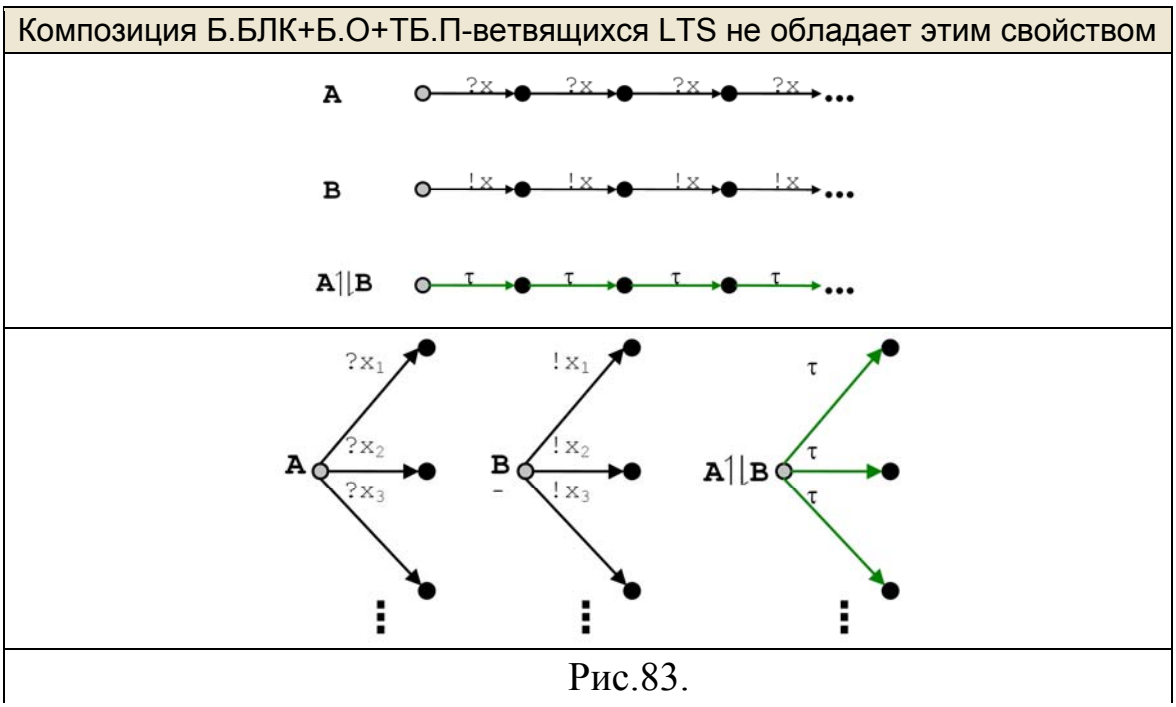


Рис.83.

Мы будем исследовать ограничения, которые нужно наложить на преобразованные LTS, чтобы их композиция была Б.БЛК+Б.О+ТБ.П-ветвящейся. Для того, чтобы преобразованные LTS удовлетворяли этим ограничениям, мы наложим соответствующие ограничения на исходные LTS. Сначала эти ограничения будут формулироваться в предположении, что алфавиты компонентов композиционной системы фиксированы. Затем эти ограничения легко переформулируются (ужесточаются) для общего случая произвольных алфавитов.

6.9.2. Ограничения на преобразованные LTS

Нам будет достаточно потребовать от преобразованной LTS и результата композиции (в том числе многократной) преобразованных LTS ЛК-ветвимости и тау-ограниченности (достижимых) неразрушающих состояний, а также тау-перечислимости любых достижимых состояний, то есть Н.ЛК+Н.О+П-ветвимости. Из такой ветвимости, очевидно, следует Б.БЛК+Б.О+ТБ.П-ветвимость. Поскольку, по Теорема 46:, для конечно-доминируемой LTS-спецификации $\mathbf{S}$ преобразованная LTS $\mathcal{T}(\mathbf{S})$ ЛК+О-ветвящаяся, она также Н.ЛК+Н.О+П-ветвящаяся. Мы покажем, что Н.ЛК+Н.О+П-ветвимость сохраняется в результате композиции, хотя ЛК+О-ветвимость может не сохраняться.

В Н.ЛК+Н.О+П-ветвящихся LTS нет маршрута с бесконечным τ-постфиксом. Поэтому в композиционной LTS он может появиться только за счёт бесконечного маршрута синхронных переходов. Поэтому естественно потребовать отсутствия бесконечного маршрута, проходящего через бесконечное число состояний, трасса которого имеет бесконечный постфикс *в синхронном алфавите*.

Также в Н.ЛК+Н.О+П-ветвящихся LTS нет бесконечного «веера» τ-переходов в неразрушающем состоянии. Поэтому в композиционной LTS он может появиться тоже только за счёт бесконечного числа синхронных переходов. Поэтому естественно потребовать отсутствия в достижимых состояниях операндов бесконечного веера переходов *по синхронным действиям*. В то же время, если один из этих синхронных переходов ведёт в состояние с γ-переходом $(s_1,s_2)\text{—}\tau\text{→}(s_1`,s_2`)\text{—}\gamma\text{→}$, то состояние (s_1,s_2) разрушающее, и нам не требуется конечность числа τ-переходов из этого состояния. Синхронный переход $(s_1,s_2)\text{—}\tau\text{→}(s_1`,s_2`)$ является композицией переходов в операндах по противоположным синхронным внешним действиям $s_1\text{—}z\text{→}s_1`$ и $s_2\text{—}\underline{z}\text{→}s_2`$. Поэтому достаточно

потребовать отсутствия в неразрушающих состояниях операндов s_1 и s_2 бесконечного веера *неразрушающих* переходов по синхронным действиям. Что касается разрушающих переходов по синхронным действиям, то достаточно перечислимости таких переходов, что гарантирует тау-перечислимость композиционного состояния.

Определение 133: Будем говорить, что LTS-модель $\mathbf{s}\in\boldsymbol{LTS}(L_\gamma)$ $L`$-*ветвящаяся*, где $L`\subseteq L$, если она Н.ЛК+Н.О+П-ветвящаяся и выполняются следующие условия:

1) в каждом достижимом состоянии определено перечислимое множество переходов суммарно по всем действиям из алфавита $L`$,
2) в каждом достижимом состоянии определено конечное число неразрушающих переходов суммарно по всем действиям из алфавита $L`$,
3) нет маршрута, проходящего через бесконечное число состояний, трасса которого имеет бесконечный постфикс в алфавите $L`$.

□

Теорема 47: Пусть заданы алфавиты $A`\subseteq A$ и $B`\subseteq B$, причём $A\cap B\subseteq A`$ и $A\cap B\subseteq B`$. Пусть также заданы $A`$-ветвящаяся LTS $\mathbf{a}\in\boldsymbol{LTS}(A_\gamma)$ и $B`$-ветвящаяся LTS $\mathbf{b}\in\boldsymbol{LTS}(B_\gamma)$. Тогда их композиция $\mathbf{c}=\mathbf{a}\!\upharpoonright\!\downharpoonright\!\mathbf{b}$ $C`$-ветвящаяся, где $C`=(A`\setminus B)\cup(B`\setminus A)$.

□591

Если композиционная система состоит из двух компонентов в алфавитах A и B, то при условии конечно-доминируемости исходных LTS-спецификаций компонентов мы должны дополнительно потребовать от преобразованных LTS-спецификаций компонентов $A\cap B$-ветвимости и $A\cap B$-ветвимости, соответственно. Тогда результат композиции будет Н.ЛК+Н.О+П-ветвящимся и, следовательно, Б.БЛК+Б.О+ТБ.П-ветвящимся.

Если композиционная система состоит из многих компонентов с заданными алфавитами и заданной схемой композиции, то для каждого

компонента соответствующим образом вычисляется тот алфавит L`, для которого компонент должен быть L`-ветвящимся. В этот алфавит попадают все действия из алфавита компонента, которые становятся синхронными после завершения композиции всей системы (отсутствуют в её алфавите). Для пояснения этого достаточно привести несколько примеров:

Пример 1: $(A \between B) \between C$.

Для компонента 1: $A \cap \underline{B} \cup (A \between B) \cap \underline{C} = A \cap \underline{B} \cup (A \setminus \underline{B}) \cap \underline{C}$

Для компонента 2: $B \cap \underline{A} \cup (A \between B) \cap \underline{C} = A \cap \underline{B} \cup (B \setminus \underline{A}) \cap \underline{C}$

Для компонента 3: $(\underline{A \between B}) \cap C = ((\underline{A} \setminus B) \cup (\underline{B} \setminus A)) \cap C$

Пример 2: $(A \between B) \between (C \between D)$.

Для компонента 1: $A \cap \underline{B} \cup (A \between B) \cap \underline{C \between D}$

$= A \cap \underline{B} \cup (A \setminus \underline{B}) \cap ((C \setminus \underline{D}) \cup (D \setminus \underline{C}))$.

Для компонента 2: $B \cap \underline{A} \cup (A \between B) \cap \underline{C \between D}$

$= B \cap \underline{A} \cup (B \setminus \underline{A}) \cap ((C \setminus \underline{D}) \cup (D \setminus \underline{C}))$.

Для компонента 3: $C \cap \underline{D} \cup (C \between D) \cap \underline{(A \between B)}$

$= C \cap \underline{D} \cup (C \setminus \underline{D}) \cap ((A \setminus \underline{B}) \cup (B \setminus \underline{A}))$.

Для компонента 4: $D \cap \underline{C} \cup (C \between D) \cap \underline{(A \between B)}$

$= D \cap \underline{C} \cup (D \setminus \underline{C}) \cap ((A \setminus \underline{B}) \cup (B \setminus \underline{A}))$.

Пример 3: $((A \between B) \between C) \between D$.

Для компонента 1: $A \cap \underline{B} \cup (A \between B) \cap \underline{C} \cup (A \between B) \between C) \cap \underline{D}$

$= A \cap \underline{B} \cup (A \setminus \underline{B}) \cap \underline{C} \cup (A \setminus \underline{B}) \setminus C) \cap \underline{D}$.

Для компонента 2: $B \cap \underline{A} \cup (A \between B) \cap \underline{C} \cup (A \between B) \between C) \cap \underline{D}$

$= A \cap \underline{B} \cup (B \setminus \underline{A}) \cap \underline{C} \cup (B \setminus \underline{A}) \setminus C) \cap \underline{D}$.

Для компонента 3: $(\underline{A \between B}) \cap C \cup (A \between B) \between C) \cap \underline{D}$

$= ((\underline{A}\backslash B)\cup(\underline{B}\backslash A))\cap C \cup (C\backslash((\underline{A}\backslash B)\cup(\underline{B}\backslash A))\cap\underline{D}.$

Для компонента 4: $\underline{(A\downarrow\uparrow B)\downarrow\uparrow C}\cap D$

$= ((A\backslash\underline{B})\cup(B\backslash\underline{A}))\backslash C \cup (\underline{C}\backslash((\underline{A}\backslash B)\cup(\underline{B}\backslash A)))\cap D.$

Если мы не фиксируем схему композиции и алфавиты операндов, то L`-ветвимость компонента, определённого в алфавите L, требуется для всего алфавита L`=L. В этом случае мы получаем конечную LTS с точностью до неразрушающих переходов.

Теорема 48: Пусть LTS $\mathbf{S}\in\boldsymbol{LTS}(L_\gamma)$ конечно-доминируемая, а для преобразованной LTS $\mathbf{T}=\mathcal{T}(\mathbf{S})$ выполнено условие 1 из Определения 133. Тогда необходимым и достаточным условием L-ветвимости LTS $\mathbf{T}$ является конечность её под-LTS $\mathbf{T}$`, порождённой всеми неразрушающими переходами и τ-переходами.

□593

Ограничения на преобразованные LTS, которые мы рассмотрели, являются достаточными для того, чтобы их композиция была Б.БЛК+Б.О+ТБ.П-ветвящейся. В общем случае они не являются необходимыми. Поскольку для конечно-доминируемой исходной LTS $\mathbf{S}$ преобразованная LTS $\mathcal{T}(\mathbf{S})$ Н.ЛК+Н.О+П-ветвящаяся, речь идёт о дополнительных условиях 1, 2 и 3 из Определения 133. В примере на Рис.84 показана преобразованная LTS, для которой не выполнено условие 3. Нарушение этого условия могло бы привести к появлению в композиции бесконечной цепочки τ-переходов. Однако для того, чтобы это произошло, необходимо, чтобы парный алфавит содержал противоположное действие $\underline{x}$, а парная LTS содержала маршрут с трассой $\langle\underline{x},\underline{x},\ldots\rangle$. Но тогда в композиции появится и трасса $\langle\gamma\rangle$, то есть в композиции не будет Б-достижимых состояний.

Условие 3 из Определения 133 не является необходимым

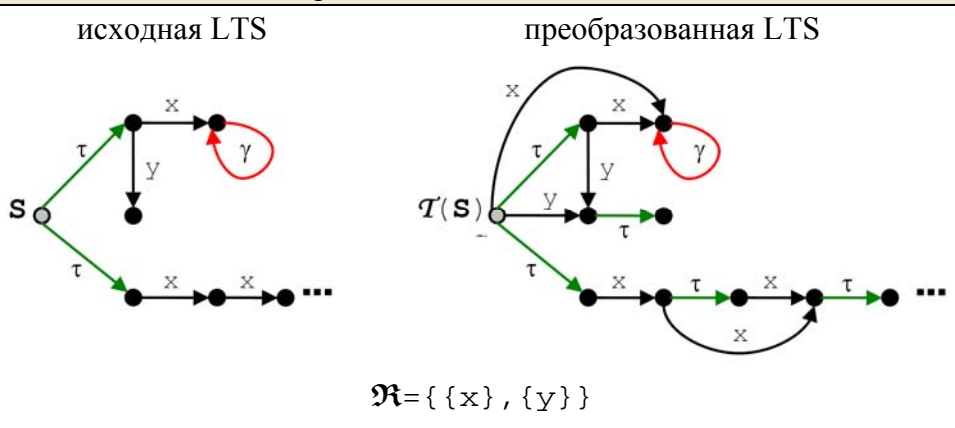


Рис.84.

6.9.3. Ограничения на исходные LTS

Ограничения на преобразованную LTS $\mathcal{T}(\mathbf{s})$ сформулированы в терминах достижимых состояний и неразрушающих переходов. Достижимому состоянию преобразованной LTS, кроме состояния γ, соответствует в исходной LTS $\mathbf{s}$ множество состояний после безопасной $\mathfrak{R}$-трассы. Неразрушающему переходу из такого состояния преобразованной LTS соответствует переход по действию, безопасному после этой $\mathfrak{R}$-трассы. Бесконечному маршруту преобразованной LTS соответствует в исходной LTS бесконечная безопасная $\mathfrak{R}$-трасса. Здесь под бесконечной безопасной $\mathfrak{R}$-трассой μ можно понимать предел бесконечно возрастающей цепочки конечных безопасных $\mathfrak{R}$-трасс. Поскольку Б.БЛК+Б.О-ветвящаяся исходная LTS $\mathbf{s}$ предельна по безопасным $\mathfrak{R}$-трассам, это эквивалентно наличию в LTS $\boldsymbol{L2L_F}(\mathbf{s})$ бесконечного маршрута с трассой μ, все конечные префиксы которой являются безопасными $\mathfrak{R}$-трассами.

Определение 134: Бесконечной $\mathfrak{R}$-трассой LTS $\mathbf{s}$ будем называть бесконечную трассу маршрута в LTS $\boldsymbol{L2L_F}(\mathbf{s})$. Бесконечную $\mathfrak{R}$-трассу будем

называть безопасной, если все её конечные префиксы являются безопасными $\mathfrak{R}$-трассами.

□

Определение 135: Будем говорить, что LTS-модель $\mathbf{s} \in \boldsymbol{LTS}(L_\gamma)$ $L`$-*трассово-ветвящаяся*, где $L` \subseteq L$, если она Б.БЛК+Б.О+ТБ.П-ветвящаяся и выполняются следующие условия:

1) каждая безопасная $\mathfrak{R}$-трасса продолжается перечислимым множеством действий из алфавита $L`$,
2) каждая безопасная $\mathfrak{R}$-трасса продолжается конечным числом безопасных после неё действий из алфавита $L`$,
3) нет бесконечной безопасной $\mathfrak{R}$-трассы, проходящей через бесконечное число состояний и имеющей бесконечный постфикс в алфавите $L`$.

□

Теорема 49: Пусть заданы вложенные алфавиты $L` \subseteq L$ и конечно-доминируемая LTS $\mathbf{s} \in \boldsymbol{LTS}(L_\gamma)$. Для того, чтобы преобразованная LTS $\mathcal{T}(\mathbf{s})$ была $L`$-ветвящаяся, необходимо и достаточно, чтобы исходная LTS $\mathbf{s}$ была $L`$-трассово-ветвящаяся.

□594

6.9.4. Алгоритмизация

При указанных ограничениях на исходные LTS преобразованные LTS и результаты их композиций будут $L`$-ветвящимися для соответствующих алфавитов $L`$. Для задания LTS-операнда композиции используются следующие алгоритмы:

Алгоритм 1: для каждого достижимого состояния и каждого действия (включая τ и γ) итератор переходов по этому действию;

Алгоритм 2: для каждого достижимого состояния итератор перечислимого множества действий из алфавита $L`$, по которым определены переходы из этого состояния.

Пусть заданы алфавиты $A` \subseteq A$ и $B` \subseteq B$, причём $A \cap \underline{B} \subseteq A`$ и $\underline{A} \cap B \subseteq B`$. Рассмотрим композицию двух алгоритмически заданных LTS: $A`$-ветвящейся LTS $\mathbf{A} \in \boldsymbol{LTS}(A_\gamma)$ и $B`$-ветвящаяся LTS $\mathbf{B} \in \boldsymbol{LTS}(B_\gamma)$. Для каждого достижимого композиционного состояния (a,b) состояния-операнды a и b также достижимы.

Композиционный Алгоритм 1 для внешнего действия z строится по аналогичному алгоритму того операнда, алфавиту которого принадлежит асинхронное действие z. Единственно, что требуется дополнительно, это оракул, определяющий, принадлежит ли данное асинхронное действие левому или правому алфавиту. Этот оракул, естественно, зависит только от алфавитов A и B.

Композиционный Алгоритм 1 для разрушения γ строится по аналогичным алгоритмам операндов: в композиционном состоянии (a,b) будет определён переход по разрушению тогда и только тогда, когда он определён в одном из состояний-операндов a или b.

Композиционный Алгоритм 1 для внутреннего действия τ должен определять как асинхронные, так и синхронные τ-переходы. Композиционные асинхронные τ-переходы строятся по τ-переходам операндов с помощью Алгоритмов 1 операндов. Для построения синхронных τ-переходов используются Алгоритмы 2 операндов, которые перечисляют, среди прочего, все синхронные действия, по которым в операндах определены переходы. Здесь используется тот факт, что $A \cap \underline{B} \subseteq A`$ и $\underline{A} \cap B \subseteq B`$. Обнаруживая синхронную пару действий $(z, \underline{z})$, мы можем по Алгоритмам 1 операндов (для действия z в левом операнде и действия $\underline{z}$ в правом операнде) построить соответствующие τ-переходы. Этот процесс заканчивается за конечное время, как только мы обнаруживаем разрушающий синхронный переход. Если все

синхронные переходы неразрушающие, то мы можем не закончить этот процесс за конечное время даже в том случае, когда число синхронных τ-переходов оказывается конечно. Проблема возникает тогда, когда в каждом состоянии a и b определены переходы по бесконечному множеству действий из алфавитов A` и B`, соответственно. Если хотя бы одно из этих множеств конечно, задача всегда решается за конечное время.

Композиционный Алгоритм 2 должен перечислять множество действий из алфавита $C`=(A`\underline{B})\cup(B`\underline{A})$, по которым определены переходы из композиционного состояния (a,b). Такой алгоритм можно построить, используя Алгоритмы 2 для операндов, перечисляющие множества действий из алфавита A` и B`, по которым определены переходы из состояний a и b, соответственно. Проблема лишь в том, что для каждого действия $z\in A`$ или $z\in B`$ мы должны определить, является он асинхронным или синхронным. Для этого нужен соответствующий оракул, который, разумеется, зависит только от алфавитов A и B.

Для конечных алфавитов все указанные проблемы легко решаются. Более того, в этом случае не требуется специальный Алгоритм 2, поскольку преобразованные LTS и результаты их композиции будут конечно-ветвящимися. Для бесконечных алфавитов могут применяться различные способы в зависимости от рассматриваемых классов LTS и/или алфавитов. Например, для βγδ-семантики подалфавит стимулов может быть бесконечным, если он разрешим относительно универсума стимулов (и задан соответствующий оракул), а подалфавит реакций конечен [27].

6.9.5. Проблема сохранения безопасности при композиции

Мы предполагаем, что всякое тестирование должно быть безопасным. При автономном тестировании безопасность реализации компонента при заданной спецификации компонента является гипотезой, которую мы вынуждены принимать на веру. Конечно, на практике могут использоваться различные

методы проверки безопасности, однако все они требуют применения специальных средств, выходящих за рамки тестирования конформности.

Если для генерации системных тестов в качестве спецификации используется косая композиция спецификаций компонентов, то нужно, чтобы безопасность реализации системы следовала из безопасности реализаций компонентов. К сожалению, в общем случае это не так: система гарантированно безопасна относительно косой композиции спецификаций компонентов только при конформных, а не просто безопасных, реализациях компонентов. Поэтому безопасность системного тестирования становится проблемой.

Сначала рассмотрим ситуацию, когда на реализации компонентов не налагается никаких ограничений, кроме их безопасности относительно спецификаций компонентов. Пусть компонуются два компонента в алфавитах A и B, имеющих LTS-спецификации $\mathbf{S}_1$ и $\mathbf{S}_2$, соответственно.

Сначала исследуем случай, когда синхронный алфавит не пуст, то есть существует некоторое синхронное действие $z \in A \between B$. На Рис.85 в строке 1 приведены примеры двух реализаций $\mathbf{I}_1$ и $\mathbf{I}_2$, которые не содержат дивергенции и разрушения. Такие реализации, очевидно, безопасны в любых $\mathfrak{R}$-семантиках для заданных алфавитов A и B любым своим LTS-спецификациям. Однако в композиции $\mathbf{I}_1 \between \mathbf{I}_2$ начальное состояние дивергентно. Такая композиция реализаций безопасна относительно спецификации системы $\mathbf{S}$ только в том случае, когда любая кнопка опасна в начале тестирования, то есть после пустой трассы спецификации системы.

Формально это означает:

$$\langle\gamma\rangle \in \mathbf{S} \vee \langle\Delta\rangle \in \mathbf{S} \vee \varnothing \notin \mathfrak{R} \;\&\; \forall x \in A \between B \;\; \langle x, \gamma\rangle \in \mathbf{S}.$$

Понятно, что никакого безопасного тестирования в этом случае происходить не будет при *любой* спецификации системы. Единственное, что мы могли бы делать, это включать машину тестирования при условии $\langle\gamma\rangle \notin \mathbf{S}$, но

никакие кнопки нажимать нельзя и, соответственно, никаких наблюдений сделать не удастся.

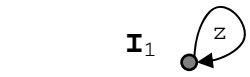
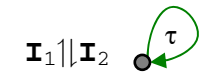
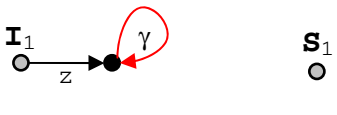
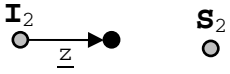
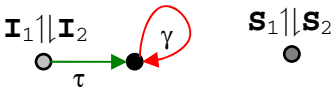
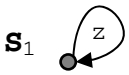
Безопасность при композиции			
	$\mathfrak{R}=\{\{z\}\}$	$\mathfrak{R}=\{\{\underline{z}\}\}$	$\mathfrak{R}=\varnothing$
1			
2	 $\mathtt{I}_1$ ***safe for*** $\mathtt{S}_1$	 $\mathtt{I}_2$ ***safe for*** $\mathtt{S}_2$	 $\mathtt{I}_1⫛\mathtt{I}_2$ ~~***safe for***~~ $\mathtt{S}_1⫛\mathtt{S}_2$
3			

Рис.85.

Более того, в некоторых случаях композиция безопасных реализаций может содержать трассу $\langle\gamma\rangle$, то есть машину тестирования нельзя даже включать (строка 2 на Рис.85). Хотя для других спецификаций такого явления не может быть (строка 3 на Рис.85).

Если синхронный алфавит пуст, то есть $A⫛\underline{B}=\varnothing$, то компоненты вообще не взаимодействуют между собой. Понятно, что в этом случае тестирование их композиции сводится к тестированию каждого из компонентов независимо от поведения другого компонента. Чтобы сгенерировать тесты для отдельного компонента, нам даже не нужно делать монотонного преобразования спецификации.

Таким образом, без дополнительных гипотез о реализациях компонентов мы либо не можем гарантировать безопасность системного тестирования, либо такое тестирование ничем не отличается от автономного тестирования отдельных компонентов. В некоторых случаях мы можем что-то «знать» о возможных реализациях компонентов, и гипотезу о безопасности системы

можно считать достаточно обоснованной. Но в других случаях требуются какие-то специальные средства.

Например, таким средством могут быть программы-«перехватчики», которые генерируются из спецификаций компонентов, перехватывают обращения между компонентами и проверяют их правильность. В терминах пред- и постусловий перехватчики проверяют выполнение предусловий операций при каждом их вызове. Тогда при системном тестировании ошибку обнаруживает либо перехватчик, либо сам системный тест. Такое тестирование уже будет безопасным. Заметим, что по спецификациям в пред- и постусловиях можно сгенерировать проверку не только пред-, но и постусловий при взаимодействии компонентов [12,60,61]. Проверка постусловий, фактически, эквивалентна вынесению вердикта системного теста, но лучше локализует ошибку: системный тест говорит, что ошибка есть в системе в целом, а перехватчик – в конкретном компоненте. Целью генерации системных тестов в этом случае становится создание надлежащих режимов взаимодействия, а все проверки выполняют перехватчики.

Следует подчеркнуть, что проблема сохранения безопасности при композиции не имеет отношения к верификации имеющейся спецификации системы с помощью косой композиции спецификаций компонентов. Также не связано с этим использование косой композиции как источника документации для пользователя системы или её разработчика, модифицирующего реализации компонентов без изменения их спецификаций и схемы компоновки. Проблема сохранения безопасности при композиции имеет отношение только к генерации тестов по косой композиции.

6.10. Выводы

В этой главе решалась проблема монотонности конформности: сохранение конформности при композиции. Суть этой проблемы в том, что композиция реализаций компонентов, конформных своим спецификациям, может оказаться неконформной композиции спецификаций компонентов, если последнюю

делать по тем же правилам, по которым делается композиция реализаций компонентов (прямая композиция):

$\mathbf{I_1}$ ***saco*** $\mathbf{S_1}$ & $\mathbf{I_2}$ ***saco*** $\mathbf{S_2}$ $\Rightarrow$ $\mathbf{I_1} \between \mathbf{I_2}$ ***saco*** $\mathbf{S_1} \between \mathbf{S_2}$.

В предыдущей главе было введено пополнение спецификаций, позволяющее перейти к семантике без $\mathfrak{Q}$-кнопок, сохраняя класс конформных и не сужая класс безопасных реализаций. В такой семантике отношение ***saco*** является предпорядком. Поэтому в данной главе рассматривались только семантики без $\mathfrak{Q}$-кнопок, а конформность считалась предпорядком.

Для решения проблемы монотонности создана общая теория монотонности, в рамках которой были введены следующие понятия. Корректная спецификация системы – это такая спецификация, которая удовлетворяет условию монотонности: прямая композиция компонентов, конформных своим спецификациям, конформна спецификации системы: $\mathbf{I_1}$ ***saco*** $\mathbf{S_1}$ & $\mathbf{I_2}$ ***saco*** $\mathbf{S_2}$ $\Rightarrow$ $\mathbf{I_1} \between \mathbf{I_2}$ ***saco*** $\mathbf{S}$. Далее была определена специальная – косая – композиция спецификаций компонентов $\mathbf{S_1} /\!\!/ \mathbf{S_2}$ как самая сильная корректная спецификация системы, разумеется, в предположении, что такая самая сильная корректная спецификация существует. Решение проблемы монотонности было предложено искать как монотонное преобразование спецификаций, обладающее тем свойством, что косая композиция спецификаций совпадает с прямой композицией монотонных (монотонно преобразованных) спецификаций: $\mathbf{S_1} /\!\!/ \mathbf{S_2} = \mathcal{T}(\mathbf{S_1}) \between \mathcal{T}(\mathbf{S_2})$.

Общая теория монотонности базируется на понятии ϕ-трасс, занимающем промежуточный уровень абстракции между моделями трасс наблюдений в $\mathfrak{R}$-семантике и LTS-моделями. В частности, для ϕ-трасс определяется оператор композиции. Сформулированы восемь достаточных условий монотонности. Шесть из них не связаны с самим монотонным преобразованием, и описывают связь между конформностью ***saco***, трассами наблюдений ($\mathfrak{R}$-трассами), ϕ-

трассами и оператором прямой композиции $\upharpoonright\!\downarrow$. Два условия налагаются на преобразование для того, чтобы оно было монотонным.

Для $\mathfrak{R}$-трасс определено отношение мажорирования и доказано его эквивалентность конформности (первое условие монотонности).

Для ϕ-трасс доказаны два условия монотонности. 1) Генеративность: по ϕ-трассам модели однозначно восстанавливаются её $\mathfrak{R}$-трассы. 2) Аддитивность: ϕ-трассы композиции LTS совпадают с композиций ϕ-трасс операндов. Первое свойство даёт возможность учитывать конформность в теории ϕ-трасс. Второе свойство позволяет рассматривать композицию, не выходя за рамки теории ϕ-трасс. Таким образом, теория ϕ-трасс оказывается замкнутой как в смысле конформности, так и в смысле композиции, что и является ключом к успеху в решении проблемы монотонности, связывающей эти два понятия: конформность и композиция.

Далее рассматриваются два монотонных преобразования. Первое преобразование определяется как множество $\boldsymbol{Conf}(\mathbf{S})$ конформных ϕ-трасс, то есть объединение ϕ-трасс всех реализаций, конформных спецификации $\mathbf{S}$. По сути, здесь доказывается существование монотонного преобразования в самом общем случае. Однако такое преобразование даёт самую большую монотонную спецификацию из всех возможных.

Остальная часть главы посвящена поиску меньших по объёму монотонных подмоделей. Для этого было введено отношение мажорирования ϕ-трасс, отвечающее двум важным условиям. 1) Генеративность: мажорирование ϕ-трасс влечёт мажорирование генерируемых ими $\mathfrak{R}$-трасс. 2) Композиционность: мажорирование ϕ-трасс операндов влечёт мажорирование ϕ-трасс композиции. Также показано, что мажорирование ϕ-трасс является предпорядком.

Такое отношение мажорирования позволяет удалить из наибольшей монотонной спецификации «лишние» ϕ-трассы, если они мажорируются

оставшимися ϕ-трассами. К сожалению, далеко не всегда можно получить наименьшую (и даже минимальную) монотонную спецификацию. Предложены три вложенные монотонные подмодели: $\boldsymbol{Conf}(\mathbf{S}) \supseteq$ финальная $\supseteq$ однородная $\supseteq$ сингулярная.

К сожалению, сингулярная монотонная подмодель $\mathcal{T}(\mathbf{S})$ наибольшей монотонной спецификации $\boldsymbol{Conf}(\mathbf{S})$ не всегда существует. Для её существования налагаются ограничения на ветвимость исходной LTS-спецификации $\mathbf{S}$. Спецификации, удовлетворяющие этим ограничениям названы конечно-доминируемыми. Частично эти ограничения совпадают с тестовыми ограничениями, то есть ограничениями, требуемыми для алгоритмической генерации тестов по исходной спецификации $\mathbf{S}$. Дополнительные ограничения нужны для того, чтобы преобразованная сингулярная модель $\mathcal{T}(\mathbf{S})$ также удовлетворяла тестовым ограничениям, то есть чтобы по ней также можно было алгоритмически генерировать тесты.

После определения монотонного преобразования изучалась проблема композиции преобразованных спецификаций. Суть её в том, что в общем случае композиция спецификаций, удовлетворяющих всем указанным ограничениям, может не удовлетворять тестовым ограничениям. Это плохо, так как в этом случае спецификация композиционной системы не годится для алгоритмической генерации системных тестов. Для решения этой проблемы определяются дополнительные ограничения на преобразованные, а затем и исходные спецификации компонентов системы. Также исследована проблема сохранения безопасности при композиции. Общий вывод такой, без дополнительных гипотез о реализациях безопасное системное тестирование либо невозможно, либо сводится к автономному тестированию отдельных компонентов.

Подводя итоги, можно сказать, что косая композиция спецификаций компонентов составной системы, выполняемая как прямая композиция монотонно преобразованных спецификаций компонентов, позволяет решить две задачи: 1) верификация согласованности имеющейся спецификации

системы со спецификациями компонентов, и 2) при отсутствии спецификации системы генерация такой спецификации по спецификациям компонентов.

Первая задача – это классическая задача верификации декомпозиции системных требований, то есть проверка правильности декомпозиции требований к системе в требования к компонентам системы. Для её решения строится косая композиция системы и проверяется её конформность имеющейся спецификации системы. Такую проверку можно делать аналитически, в том числе, моделируя тестирование косой композиции спецификаций компонентов, рассматриваемой как реализация системы, по полному набору тестов, сгенерированному из имеющейся спецификации системы.

Решение второй задачи позволяет достигнуть три цели. 1) Построенная спецификация системы может рассматриваться как описание системы с точки зрения её пользователей. 2) Такую спецификацию можно считать техническим заданием для разработчика системы, в том числе, при возможных дальнейших модификациях системы (её компонентов) с сохранением внешней функциональности и схемы компоновки. 3) По спецификации системы можно генерировать системные тесты.

Новыми являются следующие результаты этой главы:

1) Общая теория монотонности, в которой даётся формальное определение корректной спецификации, косой композиции и монотонного преобразования спецификации и сформулированы восемь достаточных условий монотонности.
2) Определение отношения мажорирования $\mathfrak{R}$-трасс и доказательство его эквивалентности безопасной конформности ***saco***.
3) Разработка концепции ϕ-трасс и модели ϕ-трасс, которая позволяет в единой трассовой теории формализовать как конформность, так и композицию, что является ключевым моментом в решении проблемы монотонности.
4) Доказательство существования монотонного преобразования спецификаций для семантики без ненаблюдаемых отказов, основанного на объединении

конформных ϕ-моделей и сохраняющего классы конформных и безопасных реализаций.

5) Определение отношения мажорирования ϕ-трасс и доказательство того, что любая конформная и мажорантная модель монотонна. Определение на этой основе вложенных монотонных подмоделей: финальной $\supseteq$ однородной $\supseteq$ сингулярной.
6) Метод монотонного преобразования спецификаций, который при достаточно слабых ограничениях на исходную спецификацию строит уменьшенную монотонную спецификацию как наибольшую сингулярную модель. Метод поддаётся алгоритмизации при достаточно слабых ограничениях на спецификацию. Частное применение этого метода решает проблему монотонности для семантики отношения ***ioco***.
7) Найдены достаточно слабые ограничения на спецификации, гарантирующие, что композиция монотонно преобразованных спецификаций удовлетворяет ограничениям алгоритмической генерации тестов. При некоторых дополнительных ограничениях эта композиция может выполняться алгоритмически.

Заключение

В настоящее время повышение качества, безопасности и надёжности ПО, как вновь создаваемого, так и давно находящегося в эксплуатации, стало важнейшей проблемой как для разработчиков реальных программных систем в индустрии, так и для исследователей в области информатики. Ключевой частью решения этой проблемы является тестирование программ, доля которого во времени и других ресурсах от общих расходов на разработку ПО составляет около 50%, а для критических по безопасности приложений – до 90%. Противоречивая задача повышения качества программ и снижения затрат на тестирование диктует необходимость поиска путей автоматизации тестирования. Одним из важнейших методов решения этой задачи является тестирование на основе формальных (математических) моделей, целью которого является проверка того, что реализационная модель (модель тестируемой системы) соответствует (конформна) спецификации (модель требований).

В диссертационной работе предлагается теория конформности для широкого класса семантик взаимодействия теста с реализацией, основанных только на наблюдаемом поведении, с параметризацией конформности типа редукции той или иной семантикой из этого класса. Область применения расширяется допущением в тестируемых реализациях ненаблюдаемых отказов, дивергенции и разрушения, моделирующего запрещённое поведение, в рамках, определяемых спецификацией. Это важно для моделирования предусловий вызова программ, для построения безопасных систем из небезопасных компонентов, для учёта дивергенции, возникающей при композиции и т.п. Для практического применения теория развита до уровня алгоритмов генерации тестов; показано, при каких ограничениях на спецификацию такие алгоритмы можно построить. И наконец, в предлагаемой теории решается проблема монотонности (сохранение конформности при композиции и при асинхронном тестировании) для рассматриваемого класса конформностей. Для этого

предлагается монотонное преобразование спецификации, которое, при определённых ограничениях, может быть реализовано алгоритмически.

Получены следующие основные результаты:

1. Метод формализации тестового эксперимента с помощью параметризуемой машины тестирования. Целью формализации является определение класса семантик взаимодействия, основанных только на наблюдениях (не на соответствии состояний) и моделирующих те или иные тестовые возможности по управлению и наблюдению поведения реализации. Также вводится понятие *разрушения*, моделирующее поведение системы, запрещённое при тестировании.
2. Введение на уровне трассовой и LTS-моделей концепции безопасного тестирования, избегающего разрушения и ненаблюдаемых отказов и не продолжающего тестовый эксперимент в том случае, когда возможна дивергенция. Введение гипотезы о безопасности реализации, которая позволяет её безопасно тестировать для заданной спецификации, и безопасной конформности типа редукции, которая опирается на гипотезу о безопасности и параметризуется семантикой взаимодействия.
3. Метод генерации полного набора тестов для безопасной конформности по спецификации, заданной в виде модели наблюдаемых трасс или LTS-модели, обобщающий аналогичные методы для некоторых частных семантик на произвольную семантику взаимодействия, основанную на наблюдаемом поведении. Метод алгоритмизуем при достаточно слабых ограничениях на спецификации (то есть ограничениях, обычно выполняемых в практических системах).
4. Метод пополнения спецификаций с использованием специальных фиктивных действий «не-отказов», который позволяет сохранять класс конформных и не сужать класс безопасных реализаций и поддаётся алгоритмизации при достаточно слабых ограничениях на спецификацию. В частности, решается проблема пополнения для семантики отношения ***ioco***.

5. Разработка концепции ϕ-трасс и модели ϕ-трасс, которые позволяют в единой трассовой теории формализовать как конформность, так и композицию, что является ключевым моментом в проблеме монотонности.
6. Метод монотонного преобразования спецификации, который при достаточно слабых ограничениях на исходную спецификацию строит монотонную спецификацию. При этом сохраняются классы безопасных и конформных реализаций, а для преобразованных спецификаций конформность сохраняется при композиции. Метод поддаётся алгоритмизации при достаточно слабых ограничениях на спецификацию. Частное применение метода решает проблему монотонности для семантики отношения ***ioco***.

Теория конформности даёт общую методологию уточнения и формализации процесса создания тестовой системы для того или иного конкретного случая. Общие алгоритмы генерации тестов и преобразования спецификаций, определяемые теорией, могут служить базовым шаблоном для разработки эффективных частных алгоритмов и реализующих их тестовых программ в практически значимых областях.

В практическом плане результаты диссертационной работы сводятся к предлагаемым методам генерации тестов и преобразования спецификаций. Это даёт, во-первых, возможность создавать тестовые системы для разных семантик тестового взаимодействия на базе единого подхода с использованием общего метода генерации тестов. Во-вторых, в сочетании с преобразованием спецификации метод применим для асинхронного тестирования с разными средами взаимодействия. В-третьих, заложена основа для построения инструментов автоматической верификации декомпозиции системных требований, причём спецификации компонентов и системы в целом могут пониматься в разных семантиках тестового взаимодействия.

В приведённой ниже таблице показаны методы тестирования, которые применялись в системах тестирования KVEST и UniTESK, разработанных в Институте Системного Программирования (ИСП РАН). Спецификации записывались в виде пред- и постусловий. Из них извлекались вручную или

автоматическим способом модели в виде автоматов или LTS. Под разрушением понималось поведение программы после обращения к ней с нарушением её предусловия. Это учитывалось при тестировании, то есть оно было безопасным. Эта таблица свидетельствует о следующем.

Во-первых, теория строилась как осмысление и обобщение того практического тестирования на основе формальных моделей, которое проводилось, в частности, в ИСП РАН.

Во-вторых, построенная теория покрывает все эти частные случаи. Конечно, общий метод генерации тестов, описанный в работе, в каждом из этих практических случаев оптимизирован с учётом указанных в таблице ограничений на реализации и спецификации.

В-третьих, теория показывает пути решения проблем тестирования для других семантик взаимодействия и других контекстов при асинхронном тестировании. Некоторые из них продемонстрированы на простейших примерах выше в разделе «Проблематика».

ТЕСТИРОВАНИЕ В ИСП РАН

Модель	Тестирование	Контекст	$\mathfrak{R}$	$\mathfrak{Q}$	Метод	Ограничения	Инструмент	Проекты
Автомат	Синхронное	–	∅	Блокировки стимулов δ	Обход графа	Детерминированный граф сильно-связный	KVEST UniTESK	Группа **1**
					Обход графа по стимулам	Недетерминиров. граф с сильно-связным полным остовным детерминиров. подграфом	UniTESK	Группа **2**
						Недетерминиров. граф сильно-Δ-связный	[UniTESK]	–
Автомат /LTS	Асинхронное	Неограниченные входные очереди и выходные очереди 1,2,… + Непосредственные реакции 0			Вызов из параллельных процессов	Дополнение к синхронному тестированию	KVEST	Группа **3**
LTS			δ_1 δ_2 …	Блокировки стимулов δ_0	Стационарное тестирование[2]	В стационарном состоянии[1] все стимулы принимаются (возможно, с разрушением)	KVEST	Группа **4**
							UniTESK	Группа **5**
	Синхронное/ Асинхронное				Нестационарное тестирование	Дополнение к стационарн. тестированию	[UniTESK]	Группа **6**

[1] *Стационарное состояние – состояние, в котором есть переходы только по стимулам (нет переходов по реакциям и τ-переходов).*
[2] *Стационарное тестирование – тестирование, при котором стимулы посылаются в реализацию только в стационарных состояниях.*

№	Название проекта	Годы	Заказчик
1	Kernel Verification	1994-8	Nortel
	FPE	1998-9	Nortel
	XA-Core	2000	Nortel
2	OLVER	2005-6	Федеральное Агентство по науке и инновациями (Роснаука)
	ОС 2000	2005-8	НИИСИ
	Java Infrastructure	2004	Intel
	Hardware Design Testing	2006-8	НИИСИ
	Germany Banking Software	2004-5	Luxoft
3	Kernel Verification	2000	Nortel
	XA-Core	2000	Nortel
	ORB	2000	Nortel
4	GWC	2000	Nortel
5	MSR IPv6	2000-1	Microsoft Research
	Mobile IPv6	2001-2	Microsoft Research, РФФИ
	OLVER	2005-6	Федеральное Агентство по науке и инновациями (Роснаука)
	ОС 2000	2005-8	НИИСИ
	Hardware Design Testing	2006-8	НИИСИ
6	Верификация распределенных систем	2003-5	Программа Президиума РАН «Разработка фундаментальных основ создания научной распределённой информационно-вычислительной среды на основе технологий GRID»
	Проблемно-ориентированные методы автоматизированной верификации распределённых систем	2006-8	

Несколько слов следует сказать о перспективах дальнейшего развития теории конформности. Во второй главе уже были указаны два таких основных направления: 1) использование трасс готовности (*ready traces*) и соответствующей безопасной конформности ***resaco***, 2) использование приоритетов.

Трассы готовности предполагают более мощные тестовые возможности и, как следствие, имеют более узкую область применимости. Но тогда, когда такие возможности есть (множества готовности наблюдаемы) и их правомерно использовать (не возникает переспецификации), тестирование по трассам готовности оправдано. Поскольку трассы готовности схожи с ϕ-трассами, можно предположить, что в рамках одной модели трасс готовности удастся соединить как определение безопасной конформности и генерацию тестов, так и решение проблемы композиции с помощью монотонного преобразования.

Использование приоритетов можно считать чрезвычайно важной и актуальной задачей ближайшего развития теории и практики конформности. Это решило бы целый ряд проблем тестирования: 1) выход из дивергенции при наличии более приоритетного «задания» для реализации, в частности, при асинхронном тестировании реактивных систем выход из цикла осцилляции (бесконечной цепочки выдачи реакций) при поступлении стимула; 2) определение альтернативного поведения реализации при возникновении тупиков (θ-переход в реализации); 3) проблема совмещения посылки стимула с приёмом реакций в реактивных системах; 4) возможность для реализации распознавать отсутствие стимулов и др.

Важное значение имеет также развитие теории конформности в сторону специальных версий теории для ограниченных классов семантик, спецификаций и реализаций. На этом пути за счёт сужения области применимости можно получить большую эффективность. Хотя это кажется возвратом к хаосу всевозможных моделей и конформностей, на самом деле этот процесс закономерен и упорядочен, поскольку идёт в рамках общей теории как её конкретизация в частных случаях с сохранением единых понятий и

методологии. В частности, весьма многообещающим кажется соединение общей теории конформности с методами тестирования конечных автоматов (в последнее время также и LTS), основанными на обходе графа переходов. Также полезные результаты получаются в области асинхронного тестирования, когда рассматриваются специальные типы среды взаимодействия, например, многоканальные входные и выходные очереди, очереди с приоритетами и т.п. По сути здесь применяются специальные разновидности монотонного преобразования спецификаций, ориентированные на класс сред. Особенностью таких тестовых систем является преобразование спецификации и композиция со средой «на лету», в процессе тестирования.

Последняя проблема, на которой хотелось бы остановиться, это проблема «разрыва» между явными моделями конечных автоматов или LTS, обычно применяемыми в теории конформности, и тем формализмом, в котором часто на практике задаются спецификации. В частности, если спецификации записываются в виде пред- и постусловий, то, хотя в основе такой спецификации лежит LTS-модель, однако она задана неявно – как решение системы уравнений. В некоторых случаях удаётся построить LTS-модель спецификации в процессе самого тестирования, точнее, ту её часть, которая оказывается необходима и достаточна для тестирования данной реализации. Этот подход оказался весьма полезным и успешно применяется в системе UniTESK. Дальнейшее развитие может быть связано с расширением области применимости этого подхода.

Литература

1. Burdonov I., Kossatchev A., Petrenko A., Cheng S., Wong H. Formal Specification and Verification of SOS Kernel. BNR/NORTEL Design Forum, June 1996.
2. Баранцев А.В., Бритвина Е.Н., Бурдонов И.Б., Косачев А.С., Гоманюк С.В., Демаков А.В., Иванов А.В., Максимов А.В., Петренко А.К., Сазанов Ю.Л., Сортов А.А., Стефанов В.П., Сумар Г.М. Архитектура системы генерации и пропуска тестов**.** Вопросы кибернетики, Москва, 1998.
3. Burdonov I., Kossatchev A., Petrenko A., Galter D. KVEST: Automated Generation of Test Suites from Formal Specifications. Proceedings of Formal Method Congress, Toulouse, France, 1999, LNCS, No. 1708.
4. Бурдонов И.Б., Косачев А.С., Демаков А.В., Петренко А.К., Максимов А.В. Формальные спецификации в технологиях обратной инженерии и верификации программ. Труды Института системного программирования РАН, No 1, 1999.
5. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Использование конечных автоматов для тестирования программ. «Программирование». 2000. No. 2.
6. Bourdonov I.B., Kossatchev A.S., Petrenko A.K., Kuliamin V.V. Experiences in using testing tools and technology in real-life applications. Proceedings of SETT'01, India, Pune, 2001.
7. Burdonov I., Kossatchev A., Demakov A., Jarov A., Petrenko A., Kuliamin V., Zelenov S. Java specification extension for automated test development. Proc. of Andrei Ershov Fourth International Conference Perspectives of System Informatics (preliminary proceedings). Novosibirsk. 2001.
8. Burdonov I., Kossatchev A., Demakov A., Jarov A., Petrenko A., Kuliamin V., Zelenov S. Java Specification Extension for Automated Test Development.

Proceedings of 4-th Intl. Conf. on Perspectives of System Informatics, LNCS 2244, Springer-Verlag, 2001.

9. Bourdonov I.B., Kossatchev A.S., Petrenko A.K., Kuliamin V.V. UniTesK Test Suite Architecture. Proceedings of 11-th Symposium on Formal Methods Europe, LNCS 2391, Springer-Verlag, 2002.

10. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Асинхронные автоматы: классификация и тестирование. Труды ИСП РАН, т. 4, 2003.

11. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай. «Программирование». 2003. No. 5.

12. Кулямин В.В., Петренко А.К., Косачев А.С., Бурдонов И.Б. Подход UniTesK к разработке тестов. «Программирование», 2003, No. 6.

13. Kuliamin V.V., Kossatchev A.S., Petrenko A.K., Pakoulin N.V., Bourdonov I.B. Integration of Functional and Timed Testing of Real-Time and Concurrent Systems. Perspectives of System Informatics // LNCS. No. 2890, Springer-Verlag, 2003.

14. Bourdonov I., Kossatchev A., Kuliamin V., Petrenko A. UniTesK: Model Based Testing in Industrial Practice. Proc of 1-st European Conference on Model-Driven Software Engineering, Nurnberg, December 2003.

15. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Неизбыточные алгоритмы обхода ориентированных графов. Недетерминированный случай. «Программирование». 2004. No. 1.

16. Бурдонов И.Б. Обход неизвестного ориентированного графа конечным роботом. «Программирование», 2004, No. 4.

17. Бурдонов И.Б. Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом. «Программирование», 2004, No. 6.

18. Бурдонов И.Б. Исследование одно/двунаправленных распределённых сетей конечным роботом. Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ". 2004.

19. Баранцев А.В., Бурдонов И.Б., Демаков А.В., Зеленов С.В., Косачев А.С., Кулямин В.В., Омельченко В.А., Пакулин Н.В., Петренко А.К., Хорошилов А.В. Подход UniTesK к разработке тестов: достижения и перспективы. Труды ИСП РАН, т. 5, М. ИСП РАН, 2004

20. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Мета-модель функциональной спецификации распределенной системы, пригодная для тестирования. Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ". 2004.

21. Бурдонов И.Б., Косачев А.С. Тестирование компонентов распределенной системы. Труды Всероссийской научной конференции «Научный сервис в сети ИНТЕРНЕТ», Изд-во МГУ, 2005.

22. Бурдонов И.Б., Косачев А.С. Верификация композиции распределенной системы. Труды Всероссийской научной конференции «Научный сервис в сети ИНТЕРНЕТ», Изд-во МГУ, 2005.

23. Bourdonov I., Kossatchev A., Kuliamin V. Formal Conformance Testing of Systems with Refused Inputs and Forbidden Actions. Proc. Of MBT 2006, Vienna, Austria, March 2006.

24. Бурдонов И.Б., Косачев А.С., Пономаренко В.Н., Шнитман В.З. Обзор подходов к верификации распределенных систем. ИСП РАН, препринт 16, М., 2006.

25. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Формализация тестового эксперимента. «Программирование», 2007, No. 5.

26. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Безопасность, верификация и теория конформности. Материалы Второй международной научной

конференции по проблемам безопасности и противодействия терроризму, Москва, МНЦМО, 2007.

27. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Теория соответствия для систем с блокировками и разрушением. «Наука», 2008.
28. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
29. Барвайс Дж. и др. Справочная книга по математической логике. Часть 2: теория множеств. М: Наука, 1982.
30. Василевский М.П. О распознавании неисправностей автомата. Кибернетика, т. 9, № 4, стр. 93–108, 1973.
31. Верещагин Н.К., Шень А. Лекции по математической логике и теории алгоритмов. Часть 1. Начала теории множеств. М.: МЦНМО, 1999.
32. Гинзбург С. Математическая теория контекстно-свободных языков. М., «МИР», 1970, 71-78.
33. Грунский И.С., Козловский В.А. Синтез и идентификация автоматов. Киев, «Наукова Думка», 2004.
34. Евтушенко Н., Вилла Т., Петренко А., Брайтон Р., Санджованни-Винцентелли А. Решение уравнений в логическом синтезе. Томский государственный университет, California University, Томск, 1999.
35. Евтушенко Н.В., Петренко А.Ф., Ветрова М.В. Недетерминированные автоматы: анализ и синтез. Часть 1, Отношения и операции. Томский государственный университет, Томск, 2006.
36. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. М. Наука, Физматлит, 1990.
37. Кудрявцев А.Б., Алешин С.В., Подколзин А.С. Введение в теорию автоматов. М.: Наука, 1985.

38. Кулямин В.В., Пакулин Н.В., Петренко О.Л., Сортов А.А., Хорошилов А.В. Формализация требований на практике. ИСП РАН, препринт 13, М., 2006.

39. Липаев В.В. Обеспечение качества программных средств. Методы и стандарты. М., СИНТЕГ, 2001.

40. Липаев В.В. Методы обеспечения качества крупномасштабных программных средств. М., СИНТЕГ, 2003.

41. Липаев В.В. Функциональная безопасность программных средств. М., СИНТЕГ, 2004.

42. Липаев В.В. Сопровождение и управление конфигурацией сложных программных средств. М., СИНТЕГ, 2006.

43. Липаев В.В. Тестирование крупных комплексов программ на соответствие требования. М., ИПЦ «Глобус», 2008.

44. Майерс Г. Искусство тестирования программ. М., Финансы и статистика, 1989.

45. Мелихов А.Н. Ориентированные графы и конечные автоматы. М. Наука, Физматлит, 1971.

46. Мур Э.Ф. Умозрительные эксперименты с последовательностными машинами. Автоматы. М.: ИЛ, 1956.

47. Общая алгебра. Под ред. Л.А.Скорнякова. т.1. М. Наука, Физматлит, 1990.

48. Оре О. Теория графов. М. Наука, Физматлит, 1968.

49. Петренко А.К. Тестирование на основе формальных спецификаций в процессах разработки программных комплексов. Диссертация на соискание уч. ст. д.ф.-м.н. М., ИСП РАН, 2003.

50. Петренко А., Бритвина Е., Грошев С., Монахов А., Петренко О. Тестирование на основе моделей. М., Открытые системы, No 9, 2003.

51. Рабин М., Скотт Д., Конечные автоматы и проблемы их разрешения. Кибернетический сборник, ИЛ, вып. 4 (1962), 58-91.

52. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений, 2-е изд., М., «Вильямс», 2002.

53. Хорошилов А.В. Спецификация и тестирование компонентов с асинхронным интерфейсом. Диссертация на соискание уч. ст. к.ф.-м.н. М., ИСП РАН, 2006.

54. Abramsky S. Observation equivalence as a testing equivalence. Theoretical Computer Science 53, 1987, pp. 225-241.

55. Aho A.V., Hopcroft J.E. The Design and Analysis of Computer Algorithms, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1974.

56. Aho A.V., Dahbura A.T., Lee D., Uyar M.Ü. An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours. IEEE Transactions on Communications, 39(11):1604–1615, 1991.

57. Alur R., Dill D.L. A theory of timed automata. Theor. Comput. Sci., 126(2):183–235, 1994.

58. Baeten J.C.M. Procesalgebra. Programmatuurkunde. Kluwer. Deventer. In Dutch. 1986.

59. Baeten J.C.M., Bergstra J.A., Klop J.W. Ready-trace semantics for concrete process algebra with the priority operator. Computer Journal 30(6), 1987, pp. 498-506.

60. Baresi L., Young M. Test oracles. Technical Report CIS-TR-01-02, University of Oregon, Deptartment of Computer and Information Science, Eugene, Oregon, U.S.A., August 2001.

61. Barnett M., Schulte W. Spying on Components: A Runtime Verification Technique. Proceedings of the OOPSLA 2001 Workshop on Specification and Verification of Component-Based Systems, 2001.

62. Beizer B. Software testing techniques. Second edition. Van Nostrand Reinholds, N-Y, 1990.

63. Bernardo M., Cleaveland R. A theory of testing for Markovian processes. In C. Palamidessi (ed), Concurrency Theory, LNCS 1877, pp. 305–319, 2000.

64. Bernot G. 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.

65. Bernot G., Gaudel M.C., Marre B. Software Testing based on Formal Specifications: a Theory and a Tool. Software Engineering Journal, v 6, n 6, November 1991, 387-405.

66. van der Bijl M., Rensink A., Tretmans J. Compositional testing with ioco. 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.

67. van der Bijl M., Rensink A., Tretmans J. Component Based Testing with ioco. CTIT Technical Report TR-CTIT-03-34, University of Twente, 2003.

68. Binder R. Testing Object-Oriented Systems: Models, Patterns, and Tools. Robert Binder. Addison-Wesley, 1999.

69. Blass A., Gurevich Y., Nachmanson L., Veanes M. 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.

70. Bloom B., Istrail S., Meyer A.R. Bisimulation can't be traced. Journal of the ACM 42(1), 1995, pp.232-268.

71. Bochmann G.V., Petrenko A. 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.

72. du Bousquet L., Ouabdesselam F., Richier J.-L., Zuanon N. Lutess: A Specification-Driven Testing Environment for Synchronous Software. ICSE 1999: 267-276.

73. Brand D. Zafiropulo P. On communicating finite-state machines. J.ACM, 30(2):323–342, 1983.

74. Brinksma E., Scollo G., Steenbergen C. LOTOS specifications, their implementations and their tests. In G. von Bochmann and B. Sarikaya, editors, Protocol Specification, Testing, and Verification VI, pages 349-360. North-Holland, 1987.

75. Brinksma E. A theory for the derivation of tests. Proc. 8th Int. Conf. Protocol Specification, Testing and Verification, North-Holland, 1988, pp. 63-74.

76. Broy M., Jonsson B., Katoen J.-P., Leucker M., Pretschner A. (Eds.). Model-Based Testing of Reactive Systems. Springer-Verlag, 2005.

77. Burton S. Automated testing from Z specifications. Technical Report (YCS 329), Department of Computer Science, University of York, 2000.

78. Chow T.S. Testing software design modeled by finite-state machines. IEEE Trans. Software Eng., vol. SE-4, no. 3, pp. 178-187, Mar. 1978.

79. Burton S, Clark J, McDermid J. Automatic generation of tests from statechart specifications. Proceedings of the First International Workshop on Formal Approaches to Testing of Software (FATES 2001), Aalborg, Denmark, August 2001.

80. Clarke D., J´eron T., Rusu V., Zinovieva E. STG: A symbolic test generation tool. In Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), number 2280 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2002.

81. Cormen T.H., Leiserson C.E., Rivest R.L. Introduction to algorithms. The MIT Press, 1991.

82. Darondeau P. An enlarged definition and complete axiomatisation of observational congruence of finite processes. In M. Dezani-Ciancaglini & U. Montanari, editors: Proceedings international symposium on programming: 5th colloquium, Aarhus, LNCS 137, Springer, 1982, pp. 47-62.

83. Dick J. Faivre A. Automating the generation and sequencing of test cases from model-based specifications. In J. C. P. Woodcock and P. G. Larsen, editors, FME'93: Industrial-Strength Formal Methods, pages 268-284. Formal Methods Europe, Springer-Verlag, Apr. 1993. Lecture Notes in Computer Science 670.

84. Edmonds J. Johnson E.L. Matching. Euler Tours, and the Chinese Postman. Math. Programming 5, 88-124 (1973).

85. Fujiwara S., Bochmann G.v., Khendek F., Amalou M., Ghedamsi A. Test Selection Based on Finite State Models. IEEE Transactions on Software Engineering, vol. 17, no. 6, pp. 591-603, Jun., 1991.

86. Fujiwara S. Bochmann G.v. 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.

87. Gargantini A., Heitmeyer C. Using model checking to generate tests from requirements specifications. Proceedings of the 7th European Software Engineering Conference, Held Jointly with the 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE’99, volume 1687 of LNCS, pages 146–162. Springer-Verlag. 1999.

88. van Glabbeek R.J., Weijland W.P. Branching time and abstraction in bisimulation semantics. Technical Report TUM-I9052, SFB-Bericht Nr. 342/29/90 A, Institut f'ür Informatik, Technische Universität München, Munich, Germany. Extended abstract in G.X. Ritter, editor: Information Processing 89, Proceedings of the IFIP 11th World Computer Congress, San Fransisco, USA 1989, Elsevier Science Publishers B,V, (North Holland), 1989, pp. 613-618.

89. van Glabbeek R.J. 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.

90. 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.

91. Goodenough J.B. Gerhart S.L. Toward a theory of test data selection. IEEE Trans. Software Eng., vol. SE-1, no. 2, pp. 156- 173, June 1975.

92. Grieskamp W., Gurevich Y., Schulte W., Veanes M. Generating Finite State Machines from Abstract State Machines. Proc. Int'l Symp. Software Testing and Analysis, Software Eng. Notes, vol. 27, no. 4, pp. 112-122, 2002.

93. Grochtmann M., Grimm K. Classification trees for partition testing. Software Testing, Verification and Reliability, 3:63-82, 1993.

94. Hartman A., Nagin K. TCBeans, software test toolkit. Proceedings of the 12th International Software Quality Week (QW99), May 1999.

95. Heerink L., Tretmans J. 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.

96. Heerink L. Ins and Outs in Refusal Testing. PhD thesis, University of Twente, Enschede, The Netherlands, 1998.

97. Hennie F.C. Fault detecting experiments for sequential circuits. Proc. 5-th Ann. Symp. Switching Circuit Theory and Logical Design, pp. 95–110, 1964.

98. Henzinger T.A. The theory of hybrid automata. Proceedings of the 11th Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, Silverspring, MD, 1996, pp. 278-292.

99. Hoare C.A.R. An axiomatic basis for computer programming. Communications of the ACM, 12(10):576–585, October 1969.

100. Hoare C.A.R. Communicating sequential processes. In R.M. McKeag & A.M. Macnaghten, editors: On the construction of programs – an advanced course, Cambridge University Press, pp/ 229-254. 1980.

101. Hoare C.A.R. Communicating Sequential Processes. Englewood Cliffs, NJ: Prentice Hall International, 1985.

102. Holzmann G.J. Design And Validation Of Computer Protocols. Prentice-Hall, 1991.

103. Huo J.L., Petrenko A. On Testing Partially Specified IOTS through Lossless Queues. Proc. Of TestCom 2004, LNCS 2978, pp. 76–94, Springer-Verlag, 2004.

104. ISO/IEC. LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization – Information Processing Systems – Open Systems Interconnection, Genéve, September 1988.

105. Jard C., Jéron T., Tanguy L., Viho C. 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.

106. Jard C., Jeron T. TGV: theory, principles and algorithms. In The Sixth World Conference on Integrated Design & Process Technology (IDPT'02), Pasadena, California, USA, June 2002.

107. Kennaway J.K. Formal semantics of nondetermism and parallelism. PhD thesis, University of Oxford, 1981.

108. Koch B., Grabowski J., Hogrefe D., Schmitt M. AutoLink – A Tool for Automatic Test Generation from SDL Specifications. Proc. IEEE Intl. Workshop on Industrial Strength Formal Specification Techniques, October 1998.

109. König D. Über eine Schlusswaise aus dem Endlichen ins Unendliche. Acta Litt. Ac. Sci. Hung. Fran. Josep. No 3. 1927. pp. 121-130. См. Также Куратовский К., Мостовский А. Теория множеств. М.«Мир», 1970.

110. Lai R. A Survey of Communication Protocol Testing. The J. Systems and Software, vol. 62, pp. 21-46, 2002.

111. Langerak R. A testing theory for LOTOS using deadlock detection. In E.Brinksma, G.Scollo, and C.A.Vissers, editors, Protocol Specification, Testing, and Verification IX, pages 87–98. North-Holland, 1990.

112. Lee D., Yannakakis M. Testing Finite State Machines: State Identification and Verification. IEEE Trans. on Computers, Vol. 43, No. 3, March 1994, pp. 306-320.

113. Lee D., Yannakakis M. Principles and Methods of Testing Finite State Machines – A Survey. Proceedings of the IEEE 84, No. 8, 1090–1123, 1996.

114. Legeard B., Peureux F., Utting M. Automated boundary testing from Z and B. In Proc. of the Int. Conf. on Formal Methods Europe, FME'02, volume 2391 of LNCS, Copenhaguen, Denmark, pages 21--40, July 2002. Springer.

115. Lestiennes G., Gaudel M.-C. Test de systemes reactifs non receptifs. Journal Europeen des Systemes Automatises, Modelisation des Systemes Reactifs, pp. 255–270. Hermes, 2005.

116. Luo G., Petrenko A., Bochmann G.v. Selecting Test Sequences for Partially Specified Nondeterministic Finite State Machines. The IFIP Seventh International Workshop on Protocol Test Systems, Japan, 1994.

117. Lynch N.A., Tuttle M.R. Hierarchical correctness proofs for distributed algorithms. Proceedings of the 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 137-151, August 1987.

118. Lynch N., Tuttle M. An Introduction to Input/Output Automata. CWI Quart, 2(3):219-246, 1989.

119. Meyer B. Applying “Design by Contract”. Computer (IEEE), vol. 25, no. 10, October 1992, pages 40-51.

120. Milner R. A Calculus of Communicating Processes. LNCS, vol. 92, Springer-Verlag, 1980.

121. Milner R. Modal characterization of observable machine behaviour. In G. Astesiano & C. Bohm, editors: Proceedings CAAP 81, LNCS 112, Springer, pp. 25-34.

122. Milner R. Communication and Concurrency. Prentice-Hall, 1989.

123. Moore E.F. Gedanken-experiments on sequential machines. C.E. Shannon (ed.), J. McCarthy (ed.), Automata studies, Princeton Univ. Press (1956), pp. 179–210.

124. Myers G. The Art of Software Testing. Wiley, 1979.

125. De Nicola R., Hennessy M.C.B. Testing equivalences for processes. Theoretical Computer Science, 34:83–133, 1984.

126. De Nicola R. Extensional equivalence for transition systems. Acta Inf., 24(2):211–237, 1987.

127. De Nicola R., Vaandrager F.W. Three logics for branching bisimulation (extended abstract). Proceedings 5th Annual Symposium on Logic in Computer Science, Philadelphia, USA, IEEE Computer Society Press, pp. 118-129. Full version available as Rapporto di Ricerca SI-92/07, Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza", November 1992.

128. De Nicola R., Segala R. A process algebraic view of Input/Output Automata. Theoretical Computer Science, 138:391-423, 1995.

129. Nielsen B. Specification and Test of Real-Time Systems. PhD thesis, Department of Computer Science, Aalborg University, Denmark, april 2000.

130. Olderog E.R., Hoare C.A.R. Specification-oriented semantics for communicating processes. Acta Informatica 23, 1986, pp. 9-66.

131. Park D. Concurrency and automata on infinite sequences. In P. Deussen, editor, Proceedings 5th GI Conference, pp. 167-183. Lecture Notes in Computer Science 104, Springer-Verlag, 1981.

132. Petrenko A., Bochmann G.v., Dssouli R. Conformance Relations and Test Derivation. The IFIP Sixth International Workshop on Protocol Test Systems, France, 1993.

133. Petrenko A., Yevtushenko N., Bochmann G.v. Testing deterministic implementations from nondeterministic FSM specifications. Selected proceedings of the IFIP TC6 9-th international workshop on Testing of communicating systems, September 1996.

134. Petrenko A., Yevtushenko N., Huo J.L. Testing Transition Systems with Input and Output Testers. Proc. IFIP TC6/WG6.1 15th Int. Conf. Testing of Communicating Systems, TestCom'2003, pp. 129-145. Sophia Antipolis, France, May 26-29, 2003.

135. Phalippou M. TVEDA: an experiment in computer-aided test case generation from formal specification of protocols. Technical Note NT/LAA/SLC/347, France Telecom - CNET, 1991.

136. Phalippou M. Relations d`Implantation et Hypotheses de Test sur des Automates a Entrees et Sorties. PhD thesis, L`Universite de Bordeaux I, France, 1994.

137. Phillips I. Refusal testing. Theoretical Computer Science, 50(2):241-284, 1987.

138. Pitt D.H., Freestone D. The derivation of conformance tests from LOTOS specifications. IEEE Transactions on Software Engineering, 16(12):1337-1343, 1990.

139. Pnueli A. Linear and branching structures in the semantics and logics of reactive systems. In W. Brauer, editor: Proceedings 12th ICALP, Nafplion, LNCS 194, Springer, 1985, pp. 15-32.

140. Pomello L. Some equivalence notions for concurrent systems – An overview. In G. Rozenberg, editor: Advances in Petri Nets 1985, LNCS 222, Springer, 1986, pp. 381-400.

141. Revised Working Draft on “Framework: Formal Methods in Conformance Testing”. JTC1/SC21/WG1/Project 54/1 // ISO Interim Meeting / ITU-T on, Paris, 1995.

142. Rouger A., Phalippou M. Test cases generation from formal specifications. In 14th International Switching Symposium, Yokohama, October 1992.

143. Segala R. Quiescence, fairness, testing, and the notion of implementation. In E. Best, editor, CONCUR'93, pages 324-338. Lecture Notes in Computer Science 715, Springer-Verlag, 1993.

144. Tan Q.M., Petrenko A., Bochmann G.v. Checking Experiments with Labeled Transition Systems for Trace Equivalence. Proceedings of the 10th International Workshop on Testing of Communicating Systems. Korea (1997) 167-182.

145. Tan Q.M., Petrenko A. Test generation for specifications modeled by input/output automata. Proceedings of the 11th International Workshop on Testing of Communicating Systems (IWTCS'98). Russia, September 1998.

146. Tretmans J. A Formal Approach to Conformance Testing. PhD. Thesis, University of Twente, Enschede, The Netherlands, 1992.

147. Tretmans J. A Formal Approach to Conformance Testing. Proceedings of the IFIP TC6/WG6.1 Sixth International Workshop on Protocol Test systems VI, p.257-276, September 28-30, 1993.

148. Tretmans J. Conformance testing with labelled transition systems: implementation relations and test generation. Computer Networks and ISDN Systems, v.29 n.1, p.49-79, Dec. 1996.

149. Tretmans J. Test Generation with Inputs, Outputs and Repetitive Quiescence. In: Software-Concepts and Tools, Vol. 17, Issue 3, 1996.

150. Vaandrager F. 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.

151. Vuong S.T., Chan W.W.L., Ito M.R. The UIOv-Method for Protocol Test Sequence Generation. Proc. Second Int'l Workshop Protocol Test Systems, pp. 161-175, 1989.

152. Walker D.J. Bisimulation and divergence. Information and Computation 85(2), 1990, pp. 202-241.

153. Wezeman C.D. The CO-OP method for compositional derivation of conformance testers. In E. Brinksma, G. Scollo, and C. A. Vissers, editors, Protocol Specification, Testing, and Verification IX, pages 145-158. North-Holland, 1990.

154. Zhu, Hall, May. Software unit test coverage and adequacy. ACM Computing Surveys, v.29, n.4, 1997.

Приложение: Доказательства утверждений

1. Доказательство Теоремы 1

1. Необходимость. Нам достаточно по дереву последовательностей Σ построить порождающий его граф, в котором все вершины конечные и одна начальная. Выберем в качестве вершин графа все пары (μ,σ), где (конечная или бесконечная) последовательность $\sigma\in\Sigma$, а $\mu\leq\sigma$ её конечный префикс. Начальной вершиной объявим дополнительную вершину ϵ, из которой проведём пустые дуги во все вершины вида (ϵ,σ). Далее проведём дугу из вершины (μ,σ) в вершину $(\mu`,\sigma)$ и пометим её символом $z$ тогда и только тогда, когда $\mu`=\mu\cdot\langle z\rangle$. По построению каждая порождаемая таким графом последовательность принадлежит дереву, и, наоборот, каждая последовательность из дерева порождается таким графом.

Заметим, что, если бы мы рассматривали только конечные последовательности, то достаточно было бы сопоставить вершинам сами последовательности дерева $\mu\in\Sigma$, а не пары (μ,σ) и проводить дугу из вершины μ в вершину $\mu`$, помеченную символом $z$, тогда и только тогда, когда $\mu`=\mu\cdot\langle z\rangle$. Так же можно было бы поступить и в том случае, когда бесконечные последовательности допускаются, но дерево замкнуто по пределу. Более сложное построение получается из-за того, что дерево может быть не замкнуто по пределу.

2. Достаточность. Нам нужно показать, что множество последовательностей, порождаемое графом, в котором все вершины конечные и одна начальная, является деревом. Это непосредственно следует из того, что все вершины дерева конечные: любой префикс последовательности, порождаемой

некоторым маршрутом графа, порождается соответствующим префиксом этого маршрута.

2. Доказательство Теоремы 2

1. $\mathfrak{R}$-проекция является деревом, поскольку деревом является ***F***-модель, а префикс $\mathfrak{R}$-трассы является $\mathfrak{R}$-трассой.

2. Допустимость и согласованность $\mathfrak{R}$-трассы не зависят от множества трасс, которому она принадлежит. Поэтому эти свойства сохраняются в $\mathfrak{R}$-проекции ***F***-модели.

3. $\mathfrak{R}$-конвергентность $\mathfrak{R}$-трассы μ, которая не содержит и не продолжается дивергенцией и разрушением, означает, что $\mathfrak{R}$-трасса μ продолжается каждым $\mathfrak{R}$-отказом или внешним действием, принадлежащим ему. Поскольку ***F***-модель $\mathcal{P}(\mathtt{L})$-конвергентна и, следовательно, $\mathfrak{R}$-конвергентна, а любое такое продолжение $\mathfrak{R}$-трассы μ оставляет её $\mathfrak{R}$-трассой, $\mathfrak{R}$-конвергентность сохраняется в $\mathfrak{R}$-проекции ***F***-модели.

4. Замкнутость означает замкнутость по ***d***-операции. Поскольку ***F***-модель замкнута, а ***d***-операция переводит $\mathfrak{R}$-трассы в $\mathfrak{R}$-трассы, замкнутость сохраняется в $\mathfrak{R}$-проекции ***F***-модели.

5. $\mathfrak{R}$-полнота означает замкнутость по ***i***-операции, то есть вставки отказа $\mathtt{Q}$ в $\mathfrak{R}$-трассу $\mu\cdot\langle\mathtt{P}\rangle\cdot\lambda$ после $\mathfrak{R}$-отказа $\mathtt{P}$ при условии, что $\mu\cdot\langle\mathtt{P}\rangle$ не продолжается ни дивергенцией, ни разрушением, ни каким-либо внешним действием из $\mathfrak{R}$-отказа $\mathtt{Q}$. Поскольку ***F***-модель замкнута по операции $\boldsymbol{i}_{\mathcal{P}(\mathtt{L})}$

и, следовательно, по операции $\boldsymbol{i}_{\mathfrak{R}}$, а после вставки $\mathfrak{R}$-отказа $\mathfrak{R}$-трасса остаётся $\mathfrak{R}$-трассой, $\mathfrak{R}$-полнота сохраняется в $\mathfrak{R}$-проекции ***F***-модели.

3. Доказательство Теоремы 3

1. Покажем, что $\boldsymbol{Ext}(\Sigma)$ дерево, если Σ дерево. Для этого достаточно показать, что при замыкании по каждой операции вместе с каждой добавляемой трассой добавляется каждый её префикс.

- Если $(\mu\cdot\lambda,\mu)\text{—}\boldsymbol{e}\rightarrow\mu\cdot\langle\varnothing\rangle\cdot\lambda$, то $(\mu\cdot\lambda`,\mu)\text{—}\boldsymbol{e}\rightarrow\mu\cdot\langle\varnothing\rangle\cdot\lambda`$ для каждого префикса $\lambda`\leq\lambda$, а префикс $\mu`\leq\mu$ был и раньше.
- Если $(\mu\cdot\lambda,\mu,Q)\text{—}\boldsymbol{i}\rightarrow\mu\cdot\langle Q\rangle\cdot\lambda$, то $(\mu\cdot\lambda`,\mu,Q)\text{—}\boldsymbol{i}\rightarrow\mu\cdot\langle Q\rangle\cdot\lambda`$ для каждого префикса $\lambda`\leq\lambda$, а префикс $\mu`\leq\mu$ был и раньше.
- Если $(\mu\cdot\langle P\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{d}\rightarrow\mu\cdot\lambda$, то $(\mu\cdot\langle P\rangle\cdot\lambda`,\mu)\text{—}\boldsymbol{d}\rightarrow\mu\cdot\lambda`$ для каждого префикса $\lambda`\leq\lambda$, а префикс $\mu`\leq\mu$ был и раньше.

2. Допустимость. Допустимость добавляемых трасс непосредственно следует из допустимости трасс дерева Σ и сохранения допустимости операциями:

- $(\mu\cdot\lambda,\mu)\text{—}\boldsymbol{e}\rightarrow\mu\cdot\langle\varnothing\rangle\cdot\lambda$.
- $(\mu\cdot\lambda,\mu,Q)\text{—}\boldsymbol{i}\rightarrow\mu\cdot\langle Q\rangle\cdot\lambda$.
- $(\mu\cdot\langle P\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{d}\rightarrow\mu\cdot\lambda$.

3. Согласованность. Согласованность добавляемых трасс непосредственно следует из согласованности трасс дерева Σ и сохранения согласованности операциями:

- $(\mu\cdot\lambda,\mu)\text{—}\boldsymbol{e}\rightarrow\mu\cdot\langle\varnothing\rangle\cdot\lambda$: Трасса $\mu\cdot\langle\varnothing\rangle\cdot\lambda$ согласована, поскольку $\mu\cdot\lambda$ согласована и ***e***-операция требует $\lambda\notin\{\Delta,\gamma\}$.

- $(\mu\cdot\lambda,\mu,Q)\text{—}\boldsymbol{i}\text{→}\mu\cdot\langle Q\rangle\cdot\lambda$: Трасса $\mu\cdot\langle Q\rangle\cdot\lambda$ согласована, поскольку $\mu\cdot\lambda$ согласована и ***i***-операция требует $\lambda\notin\{\langle u\rangle\mid u\in Q_{\Delta\gamma}\}$.
- $(\mu\cdot\langle P\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{d}\text{→}\mu\cdot\lambda$: Трасса $\mu\cdot\lambda$ согласована, поскольку $\mu\cdot\langle P,Q\rangle\cdot\lambda$ согласована.

4. $\mathcal{P}(L)$-конвергентность. Конвергентность трасс следует из того, что при замыкании для каждой трассы добавляются нужные продолжения:

- После замыкания по ***e***-операции все трассы, которые не заканчиваются и не продолжаются дивергенцией, разрушением и непустыми отказами, продолжаются пустым отказом, то есть конвергентны по пустому отказу.
- Далее после замыкания по ***i***-операции все трассы, которые заканчиваются отказом, $\mathcal{P}(L)$-конвергентны. Неконвергентной может быть только трасса σ, которая не заканчиваются на отказ. Но, если σ не заканчивается и не продолжается дивергенцией и разрушением, то она продолжается каким-нибудь отказом $P\subseteq L$ (на который заканчивается трасса или пустым отказом), после которого трасса $\sigma\cdot\langle P\rangle$ уже $\mathcal{P}(L)$-конвергентна.
- Далее замыкание по ***d***-операции делает все трассы $\mathcal{P}(L)$-конвергентными по любому отказу P.

5. Замкнутость. Замкнутость дерева $\boldsymbol{Ext}(\boldsymbol{\Sigma})$ непосредственно следует из того, что оно получается из дерева замыканием по ***d***-операции.

6. $\mathcal{P}(L)$-полнота. После замыкания по ***i***-операции получается дерево, замкнутое по ***i***-операции. Нам нужно лишь показать, что эта замкнутость не нарушается далее при замыкании по ***d***-операции.

Поскольку мы добавляем трассы, но не удаляем их, нарушение полноты может произойти только после добавленной трассы, заканчивающейся отказом. Пусть $(\mu\cdot\langle P\rangle\cdot\mu`\cdot\langle P`\rangle\cdot\lambda,\mu)\text{—}\boldsymbol{d}\text{→}\mu\cdot\mu`\cdot\langle P`\rangle\cdot\lambda$. Если трасса

$\mu\cdot\mu`\cdot\langle P`\rangle$ не продолжается действиями из отказа Q, то до замыкания трасса $\mu\cdot\langle P\rangle\cdot\mu`\cdot\langle P`\rangle$ также не продолжалась этими действиями (это доказывается от противного, когда λ состоит из одного такого действия). Значит вместе с любой трассой $\mu\cdot\langle P\rangle\cdot\mu`\cdot\langle P`\rangle\cdot\lambda$ была трасса $\mu\cdot\langle P\rangle\cdot\mu`\cdot\langle P`,Q\rangle\cdot\lambda$, а тогда после замыкания вместе с любой трассой $\mu\cdot\mu`\cdot\langle P`\rangle\cdot\lambda$ будет трасса $\mu\cdot\mu`\cdot\langle P`,Q\rangle\cdot\lambda$.

7. Сохранение множества R-трасс: $\boldsymbol{Ext}(\boldsymbol{\Sigma})^{\downarrow}L_{\mathfrak{R}\Delta\gamma}=\boldsymbol{\Sigma}$. Если семейство $\mathfrak{R}$ не содержит пустого отказа, ***e***-операция не меняет множество $\mathfrak{R}$-трасс. А если семейство $\mathfrak{R}$ содержит пустой отказ, ***e***-операция совпадает с тождественным преобразованием. После этого ***di***-операции, очевидно, также не меняют множество $\mathfrak{R}$-трасс.

4. Доказательство Теоремы 4

1. Объединение деревьев трасс является деревом.

2. Допустимость и согласованность являются свойствами трасс независимо от того, какому множеству трасс они принадлежат. Поэтому, поскольку эти свойства выполнены в каждой из объединяемых ***F***-моделей, они выполнены в объединении.

3. $\mathcal{P}(L)$-конвергентность. Если трасса объединения не заканчивается и не продолжается дивергенцией и разрушением в объединении, то она не заканчивается и не продолжается дивергенцией и разрушением в той (быть может, не единственной) ***F***-модели $\boldsymbol{\Sigma}$, которой она принадлежит. Следовательно, эта трасса $\mathcal{P}(L)$-конвергентна в $\boldsymbol{\Sigma}$. Очевидно, такая трасса

$\mathcal{P}(\mathtt{L})$-конвергентна в любом надмножестве трасс, в частности, в объединении ***F***-моделей.

4. Замкнутость. ***d***-замыкание трассы объединения принадлежит той ***F***-модели (быть может, не единственной), которой принадлежит сама трасса. Тем более оно принадлежит любому надмножеству трасс, в частности, объединению ***F***-моделей.

5. $\mathcal{P}(\mathtt{L})$-полнота. Пусть трасса объединения имеет вид $\mu\cdot\langle \mathtt{P}\rangle\cdot\lambda$, где $\mathtt{P}\subseteq\mathtt{L}$ отказ, и трасса $\mu\cdot\langle \mathtt{P}\rangle$ не продолжается в объединении дивергенцией и разрушением и не продолжается ни одним внешним действием из отказа $\mathtt{Q}\subseteq\mathtt{L}$. Трасса $\mu\cdot\langle \mathtt{P}\rangle\cdot\lambda$ принадлежит некоторой (быть может, не единственной) ***F***-модели $\boldsymbol{\Sigma}$. Очевидно, в ней трасса $\mu\cdot\langle \mathtt{P}\rangle$ также не продолжается дивергенцией и разрушением и не продолжается ни одним внешним действием из отказа $\mathtt{Q}$. По $\mathcal{P}(\mathtt{L})$-полноте ***F***-модели $\boldsymbol{\Sigma}$, в ней имеется трасса $\mu\cdot\langle \mathtt{P},\mathtt{Q}\rangle\cdot\lambda$. А тогда такая трасса есть в любом надмножестве трасс, в частности, в объединении ***F***-моделей.

5. Доказательство Теоремы 5

Обозначим через $\boldsymbol{\Sigma}$ множество наблюдаемых трасс $\mathfrak{R}/\mathfrak{Q}$-машины тестирования.

1. Множество $\boldsymbol{\Sigma}$ является деревом, поскольку наблюдаемость трассы означает также наблюдаемость любого её префикса.

2. Допустимость. Поскольку после дивергенции и разрушения в машине дальнейшие наблюдения невозможны, они могут быть только последними символами наблюдаемых трасс.

3. Согласованность. Ниже мы докажем замкнутость $\boldsymbol{\Sigma}$, поэтому нам достаточно доказать свойство ослабленной согласованности: после отказа не может

наблюдаться дивергенция, разрушение или внешнее действие из этого отказа. Наблюдение отказа P возникает при нажатии $\mathfrak{R}$-кнопки "P", когда машина стоит. Для машины без приоритетов это означает, что нет дивергенции и разрушения, и каждое внешнее действие, разрешаемое кнопкой "P", не определено, то есть они не могут наблюдаться после отказа "P".

4. $\mathfrak{R}$-конвергентность. Пусть некоторая наблюдаемая трасса σ не заканчивается дивергенцией и разрушением. Если после наблюдения трассы σ машина не может остановиться при отсутствии нажатых кнопок, то в каждый момент времени определено разрушение или внутреннее действие. Если разрушения нет, значит имеется бесконечная цепочка внутренних действий, то есть дивергенция. Таким образом, если трасса σ не заканчивается и не продолжается дивергенцией и разрушением, то после трассы σ машина может остановиться. А тогда после остановки при нажатии любой $\mathfrak{R}$-кнопки "P" должно наблюдаться либо внешнее действие из P, либо отказ P, то есть трасса P-конвергентна.
5. Замкнутость. Пусть наблюдается трасса $\mu\cdot\langle P\rangle\cdot\lambda$, где P $\mathfrak{R}$-отказ. В машине без приоритетов это значит, что машина может остановиться после трассы μ, причём все внешние действия, разрешаемые кнопкой "P" не определены. Отсюда непосредственно следует замкнутость по операции удаления отказа: для наблюдения отказа P после трассы μ мы нажимали кнопку "P", а если этого не делать, то мы сможем наблюдать трассу $\mu\cdot\lambda$.
6. $\mathfrak{R}$-полнота. Если для любого внешнего действия $z\in Q$, где Q $\mathfrak{R}$-отказ, не наблюдаема трасса $\mu\cdot\langle P, z\rangle$, где P $\mathfrak{R}$-отказ, то после такой остановки машины после трассы μ, когда возможно наблюдение отказа P, не определены действия из Q. Поэтому, если бы мы вставили после наблюдения

трассы $\mu\cdot\langle P\rangle$ нажатие кнопки “Q”, мы смогли бы наблюдать трассу $\mu\cdot\langle P, Q\rangle\cdot\lambda$.

6. Доказательство Леммы 1

Имеем $O=\mathit{head}(\mathbf{T})\cap\mathcal{P}(L)$.

1. Сначала покажем, что $O^*\subseteq\mathbf{T}$.

Будем вести доказательство от противного: пусть есть трасса $\rho\in O^*\setminus\mathbf{T}$.

Выберем трассу ρ минимальной по длине среди всех таких трасс.

Трасса ρ не пуста, так как пустая трасса принадлежит любому дереву.

Тогда трассу можно представить в виде $\rho=\rho`\cdot\langle r\rangle$, где трасса $\rho`\in O^*\cap\mathbf{T}$, а отказ $r\in O\setminus\mathit{head}(\mathbf{T}\ \mathit{after}\ \rho`)$.

По свойству (стаб.1), $\mathbf{T}\ \mathit{after}\ \rho` = \mathbf{T}$.

Следовательно,

$O\setminus\mathit{head}(\mathbf{T}\ \mathit{after}\ \rho`)$

$= O\setminus\mathit{head}(\mathbf{T})$

$= (\mathit{head}(\mathbf{T})\cap\mathcal{P}(L))\setminus\mathit{head}(\mathbf{T})$

$= \varnothing$, чего быть не может, поскольку $r\in O\setminus\mathit{head}(\mathbf{T}\ \mathit{after}\ \rho`)$.

2. Теперь покажем, что любая трасса ρ отказов из $\mathbf{T}$ принадлежит O^*.

Будем вести доказательство от противного: пусть есть трасса $\rho\in\mathbf{T}\setminus O^*$.

Выберем трассу ρ минимальной по длине среди всех таких трасс.

Трасса ρ не пуста, так как $\epsilon\in O^*$.

Тогда трассу можно представить в виде $\rho=\rho`\cdot\langle r\rangle$, где $\rho`\in\mathbf{T}\cap O^*$, а отказ $r\in(\mathit{head}(\mathbf{T}\ \mathit{after}\ \rho`)\cap\mathcal{P}(L))\setminus O$.

По свойству (стаб.1), $\mathbf{T}\ \mathit{after}\ \rho` = \mathbf{T}$.

Следовательно

$(\mathit{head}(\mathbf{T}\ \mathit{after}\ \rho`)\cap\mathcal{P}(L))\setminus O$

$= (\mathit{head}(\mathbf{T})\cap\mathcal{P}(L))\setminus O$

$= O\setminus O$

$= \varnothing$, что противоречит $r\in(\mathit{head}(\mathbf{T}\ \mathit{after}\ \rho`)\cap\mathcal{P}(L))\setminus O$.

3. Покажем, что $\mathbf{T}\subseteq O^*\cdot\mathbf{T}_c$.

Любая трасса из $\mathbf{T}$ может быть представлена в виде $\rho\cdot\lambda\in\mathbf{T}$, где $\rho\in\mathcal{P}(L)^*$, а трасса λ не начинается с отказа.

Поскольку любая трасса отказов из $\mathbf{T}$ принадлежит O^*, имеем $\rho\in O^*$.

$\rho\cdot\lambda\in\mathbf{T}$ влечёт $\lambda\in(\mathbf{T}\ \mathit{after}\ \rho)$.

По свойству (стаб.1), $\mathbf{T}\ \mathit{after}\ \rho = \mathbf{T}$.

Поэтому $\lambda\in\mathbf{T}$ и, поскольку λ не начинается с отказа, $\lambda\in\mathbf{T}_c$.

Тем самым, $\rho\cdot\lambda\in O^*\cdot\mathbf{T}_c$.

4. Покажем, что $O^*\cdot\mathbf{T}_c\subseteq\mathbf{T}$.

Пусть трасса $\rho\cdot\lambda\in O^*\cdot\mathbf{T}_c$, где $\rho\in O^*$, $\lambda\in\mathbf{T}_c$.

Поскольку $O^*\subseteq\mathbf{T}$, то $\rho\in\mathbf{T}$.

Поскольку $\lambda\in\mathbf{T}_c$, а $\mathbf{T}_c\subseteq\mathbf{T}$, то $\lambda\in\mathbf{T}$.

По свойству (стаб.1), $\mathbf{T}\ \mathit{after}\ \rho = \mathbf{T}$.

Значит, $\lambda\in(\mathbf{T}\ \mathit{after}\ \rho)$.

Тем самым, $\rho\cdot\lambda\in\mathbf{T}$.

7. Доказательство Леммы 2

1. Допустимость. Если трасса $\mu\cdot\sigma\in\mathbf{\Sigma}$ допустима (не содержит Δ и γ «внутри»), то очевидно, её постфикс $\sigma\in(\mathbf{\Sigma}\ \mathit{after}\ \mu)$ также допустим.

2. Согласованность. Если в трассе $\mu\cdot\sigma\in\boldsymbol{\Sigma}$ за отказом не следует дивергенция, разрушение или внешнее действие из этого отказа, то, очевидно, это верно и для её постфикса $\sigma\in(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$.

3. $\mathcal{P}(\mathtt{L})$-конвергентность. Если трасса $\sigma\in(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$ не заканчивается и не продолжается дивергенцией и разрушением в дереве $\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu$, то, очевидно, трасса $\mu\cdot\sigma\in\boldsymbol{\Sigma}$ не заканчивается и не продолжается дивергенцией и разрушением в дереве $\boldsymbol{\Sigma}$. По $\mathcal{P}(\mathtt{L})$-конвергентности дерева $\boldsymbol{\Sigma}$, трасса $\mu\cdot\sigma\in\boldsymbol{\Sigma}$ продолжается в дереве $\boldsymbol{\Sigma}$ каждым отказом или внешним действием из этого отказа. Но тогда это верно и для её постфикса $\sigma\in(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$ в дереве $\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu$.

4. Замкнутость. Для любой трассы $\sigma\in(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$ имеет место: $\{\mu\}\cdot\boldsymbol{d}(\sigma)\subseteq\boldsymbol{d}(\mu\cdot\sigma)$. Поскольку $\mu\cdot\sigma\in\boldsymbol{\Sigma}$, по замкнутости дерева $\boldsymbol{\Sigma}$, $\boldsymbol{d}(\mu\cdot\sigma)\subseteq\boldsymbol{\Sigma}$. Поэтому $\{\mu\}\cdot\boldsymbol{d}(\sigma)\subseteq\boldsymbol{\Sigma}$, что влечёт $\boldsymbol{d}(\sigma)\subseteq(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$.

5. $\mathcal{P}$(L)-полнота. Пусть трасса $\sigma\cdot\lambda\in(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$, где σ заканчивается отказом. Тогда трасса $\mu\cdot\sigma$ в дереве $\boldsymbol{\Sigma}$ также заканчивается отказом. Если трасса σ в дереве $\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu$ не продолжается ни одним внешним действием из отказа $\mathtt{Q}$, то трасса $\mu\cdot\sigma$ в дереве $\boldsymbol{\Sigma}$ также не продолжается ни одним внешним действием из отказа $\mathtt{Q}$. А тогда, по $\mathcal{P}$(L)-полноте дерева $\boldsymbol{\Sigma}$, между трассами $\mu\cdot\sigma$ и λ можно вставить отказ $\mathtt{Q}$, оставаясь в рамках дерева $\boldsymbol{\Sigma}$. Но тогда, очевидно, это можно сделать между трассами σ и λ, оставаясь в рамках дерева $\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu$.

8. Доказательство Леммы 3

Пусть заданы алфавит $\mathtt{L}\subseteq\mathtt{Z}$ и ***F***-модель $\boldsymbol{\Sigma}\in\boldsymbol{FMODEL}(\mathtt{L})$.

Наибольшее контрстабильное поддерево состоит из всех трасс ***F***-модели $\boldsymbol{\Sigma}$, начинающихся не с отказов:

$\mathbf{\Sigma}_c = \{\mu \in \mathbf{\Sigma} \mid \textit{pref}(\mu) = \epsilon\} = \{\epsilon\} \cup \{\langle z \rangle \cdot \lambda \in \mathbf{\Sigma} \mid z \in L_{\Delta\gamma} \ \&\ \lambda \in L_{\mathcal{P}(L)\Delta\gamma}{}^*\}$.

1. Сначала покажем, что $\mathbf{\Sigma}_c$ после каждого внешнего действия остаётся ***F***-моделью.

Очевидно, $\forall z \in L \ \ \mathbf{\Sigma}_c \textit{ after } \langle z \rangle = \mathbf{\Sigma} \textit{ after } \langle z \rangle$.

По Лемме 2, $\mathbf{\Sigma} \textit{ after } \langle z \rangle$ является ***F***-моделью.

Следовательно, $\mathbf{\Sigma}_c \textit{ after } \langle z \rangle$ является ***F***-моделью.

2. Теперь покажем, что в $\mathbf{\Sigma}_c$ дивергенция и разрушение ничем не продолжаются.

Очевидно, $\langle \Delta \rangle \cdot \lambda \in \mathbf{\Sigma}_c$ влечёт $\langle \Delta \rangle \cdot \lambda \in \mathbf{\Sigma}$, а $\langle \gamma \rangle \cdot \lambda \in \mathbf{\Sigma}_c$ влечёт $\langle \gamma \rangle \cdot \lambda \in \mathbf{\Sigma}$.

По допустимости $\mathbf{\Sigma}$, $\langle \Delta \rangle \cdot \lambda \in \mathbf{\Sigma}$ влечёт $\lambda = \epsilon$, $\langle \gamma \rangle \cdot \lambda \in \mathbf{\Sigma}$ влечёт $\lambda = \epsilon$.

Следовательно, $\langle \Delta \rangle \cdot \lambda \in \mathbf{\Sigma}_c$ влечёт $\lambda = \epsilon$, $\langle \gamma \rangle \cdot \lambda \in \mathbf{\Sigma}_c$ влечёт $\lambda = \epsilon$.

Из 1 и 2 следует, что $\mathbf{\Sigma}_c$ является квази-моделью.

9. Доказательство Леммы 4

Имеем $O = \mathcal{P}(L \setminus \textit{head}(\mathbf{T}))$. Очевидно, $O \neq \varnothing$ (содержит пустой отказ). Поскольку $\langle \Delta \rangle \notin \mathbf{T}$ и $\langle \gamma \rangle \notin \mathbf{T}$, любая трасса из $O^* \cdot \mathbf{T}$ имеет вид ρ или $\rho \cdot \langle z \rangle \cdot \sigma$, где $\rho \in O^*$, $z \in L$ и $\langle z \rangle \cdot \sigma \in \mathbf{T}$.

1. Допустимость. Трасса ρ, как любая трасса отказов, допустима. По допустимости дерева $\mathbf{T} \textit{ after } \langle z \rangle$, трасса σ допустима (не содержит Δ и γ «внутри»). Тогда конкатенация $\rho \cdot \langle z \rangle \cdot \sigma$ также допустима.

2. Согласованность. Трасса ρ, как любая трасса отказов, согласована. Поскольку дерево $\mathbf{T} \textit{ after } \langle z \rangle$ согласовано, трасса σ согласована.

Поскольку $\forall P \in O = \mathcal{P}(L \setminus \mathit{head}(\mathbf{T}))$ $z \notin P$, конкатенация $\rho \cdot \langle z \rangle \cdot \sigma$ согласована.

3. $\mathcal{P}(L)$-конвергентность. Трасса ρ $\mathcal{P}(L)$-конвергентна, поскольку $\mathit{head}(O^* \cdot \mathbf{T}\ \mathit{after}\ \rho) = O \cup \mathit{head}(\mathbf{T})$, а это множество, по определению O, содержит пустой отказ, а также каждый непустой отказ или, по крайней мере, одно внешнее действие из него. Поскольку $\mathbf{T}\ \mathit{after}\ \langle z \rangle$ $\boldsymbol{F}$-модель, трасса σ, не заканчивающаяся и не продолжающаяся дивергенцией и разрушением, $\mathcal{P}(L)$-конвергентна в $\mathbf{T}\ \mathit{after}\ \langle z \rangle$. Тем самым, конкатенация $\rho \cdot \langle z \rangle \cdot \sigma$ $\mathcal{P}(L)$-конвергентна в $\mathbf{T}$.

4. Замкнутость. $\boldsymbol{d}(\rho) \subseteq O^* \subseteq O^* \cdot \mathbf{T}$. По замкнутости $\mathbf{T}\ \mathit{after}\ \langle z \rangle$,

$\boldsymbol{d}(\rho \cdot \langle z \rangle \cdot \sigma) = \boldsymbol{d}(\rho) \cdot \{\langle z \rangle\} \cdot \boldsymbol{d}(\sigma) \subseteq O^* \cdot \{\langle z \rangle\} \cdot (\mathbf{T}\ \mathit{after}\ \langle z \rangle) \subseteq O^* \cdot \mathbf{T}$.

5. $\mathcal{P}(L)$-полнота. По определению O, трасса ρ не продолжается внешними действиями из отказа Q только в том случае, когда $Q \in O$. В этом случае для любой трассы $\rho \cdot \lambda \in O^* \cdot \mathbf{T}$ трасса $\rho \cdot \langle Q \rangle \cdot \lambda \in O^* \cdot \mathbf{T}$. Поскольку дерево $\mathbf{T}\ \mathit{after}\ \langle z \rangle$ $\mathcal{P}(L)$-полно, $\mathfrak{R}$-полнота в нём его трассы σ влечёт выполнение этого свойства для трассы $\rho \cdot \langle z \rangle \cdot \sigma$ в дереве $O^* \cdot \mathbf{T}$.

10. Доказательство Теоремы 6

1. По Лемме 3, наибольшее контрстабильное дерево $\boldsymbol{\Sigma}_C$ является квази-моделью. Все трассы $\lambda \in \boldsymbol{\Sigma}$, не начинающиеся с отказа, принадлежат дереву $\boldsymbol{\Sigma}_C$, и все трассы дерева $\boldsymbol{\Sigma}_C$ принадлежат $\boldsymbol{\Sigma}$.

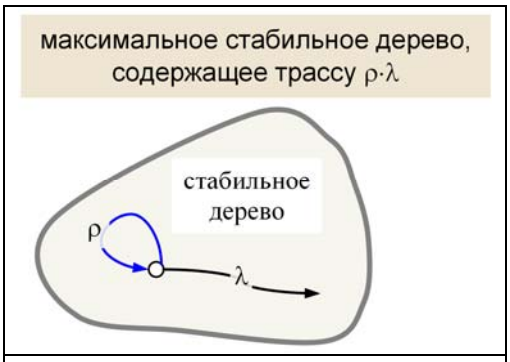


Рис.86.

2. Пусть трасса $\rho\cdot\lambda\in\mathbf{\Sigma}$, где ρ – непустая трасса отказов, а трасса λ не начинается с отказа: $\rho=\mathit{pref}(\rho\cdot\lambda)\neq\epsilon$. Для этой трассы мы построим максимальное стабильное поддерево $\mathbf{T}\subseteq\mathbf{\Sigma}$, содержащее эту трассу $\rho\cdot\lambda\in\mathbf{T}$, и такое, что его наибольшее контрстабильное поддерево совпадает с наибольшим контрстабильным поддеревом дерева $\mathbf{\Sigma}$ после трассы отказов ρ: $\mathbf{T}_c=(\mathbf{\Sigma}\ \mathit{after}\ \rho)_c$. Заметим, что, по свойству согласованности, $(\mathbf{\Sigma}\ \mathit{after}\ \rho)_c$ не содержит трасс $\langle\Delta\rangle$ и $\langle\gamma\rangle$. (см. Рис.86)

2.1. Обозначим $O=\mathit{Im}(\rho)$. По свойству ***drt***-замкнутости дерева $\mathbf{\Sigma}$, любая трасса μ, продолжающая в $\mathbf{\Sigma}$ трассу ρ, продолжает в $\mathbf{\Sigma}$ каждую трассу из O^*: $O^*\cdot(\mathbf{\Sigma}\ \mathit{after}\ \rho)\subseteq\mathbf{\Sigma}$.

Обозначим $\mathbf{T}(O)=O^*\cdot(\mathbf{\Sigma}\ \mathit{after}\ \rho)$.

2.1.1. Очевидно, $\rho\cdot\lambda\in\mathbf{T}(O)$.

2.1.2. По построению, $\mathbf{T}(O)$ – дерево.

2.1.3. Имеем $\mathbf{T}(O)_c=(O^*\cdot(\mathbf{\Sigma}\ \mathit{after}\ \rho))_c=(\mathbf{\Sigma}\ \mathit{after}\ \rho)_c$.

2.1.4. По построению, $\forall\rho`\in O^*\ \mathbf{T}(O)\ \mathit{after}\ \rho`=\mathbf{T}(O)$, то есть, почти выполнено свойство (стаб.1): оно может быть не выполнено только для трассы отказов $\rho`\notin O^*$.

2.1.5. Так как ρ – непустая трасса отказов, то, по согласованности дерева $\mathbf{\Sigma}$, трасса ρ не продолжается дивергенцией и разрушением. А тогда, по $\mathcal{P}(L)$-конвергентности модели $\mathbf{\Sigma}$, трасса ρ продолжается каждым отказом или внешним действием из него. Отсюда следует следующее свойство для O^*:

$\forall Q\subseteq L\ \ O^*\cdot\{\langle Q\rangle\}\subseteq\mathbf{T}(O)\ \vee\ \exists z\in Q\ \ O^*\cdot\{\langle z\rangle\}\subseteq\mathbf{T}(O)$.

Поскольку $\mathbf{T}(O)$ ***after*** $\rho = \mathbf{T}(O)$, это свойство эквивалентно следующему свойству:

$\forall Q \subseteq L \ \langle Q \rangle \in \mathbf{T}(O) \ \vee \ \exists z \in Q \ \langle z \rangle \in \mathbf{T}(O)$.

Поскольку $\mathbf{T}(O)$ ***after*** $\rho = \mathbf{T}(O)$, а трасса ρ не продолжается дивергенцией и разрушением, $\langle \Delta \rangle \notin \mathbf{T}(O)$ и $\langle \gamma \rangle \notin \mathbf{T}(O)$.

Таким образом, выполнено свойство (стаб.2).

Мы видим, что дерево $\mathbf{T}(O)$ содержит исходную трассу $\rho \cdot \lambda$, $\mathbf{T}(O)_c = (\mathbf{\Sigma}$ ***after*** $\rho)_c$, 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^*.

2.2. 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). (Смотри пример на Рис.87 для βγδ-модели, блокировок и стационарности.)

«Недостающий» отказ – это такой отказ $Q \notin O$, что трассы отказов из O не продолжаются внешними действиями из Q и продолжаются отказом Q:

Plus$(O) = \{ \ Q \in \mathcal{P}(L) \setminus O \mid \langle Q \rangle \in \mathbf{T}(O) \ \& \ \forall z \in Q \ \langle z \rangle \notin \mathbf{T}(O) \ \}$.

Определим $O` = O \cup$ ***Plus***(O).

По свойству $\mathcal{P}(L)$-полноты модели $\mathbf{\Sigma}$, $\mathbf{T}(O`) = \mathbf{T}(O)$.

(Заметим, что в дереве $\mathbf{\Sigma}$ верно $O``^* = \cup_\circ \boldsymbol{i}(O^*) = \cup_\circ$ ***drti***$(O^*) =$ ***drti***(ρ).)

Для $\mathbf{T}(O`)$, по-прежнему, $\rho \cdot \lambda \in \mathbf{T}(O`)$, $\mathbf{T}(O`)_c = (\mathbf{\Sigma}$ ***after*** $\rho)_c$, выполнено свойство (стаб.2) и почти выполнено свойство (стаб.1): $\forall \rho` \in O``^* \ \mathbf{T}(O`)$ ***after*** $\rho` = \mathbf{T}(O`)$.

Но для $\mathbf{T}(O`)$ множество «недостающих» отказов ***Plus***$(O`)$ пусто.

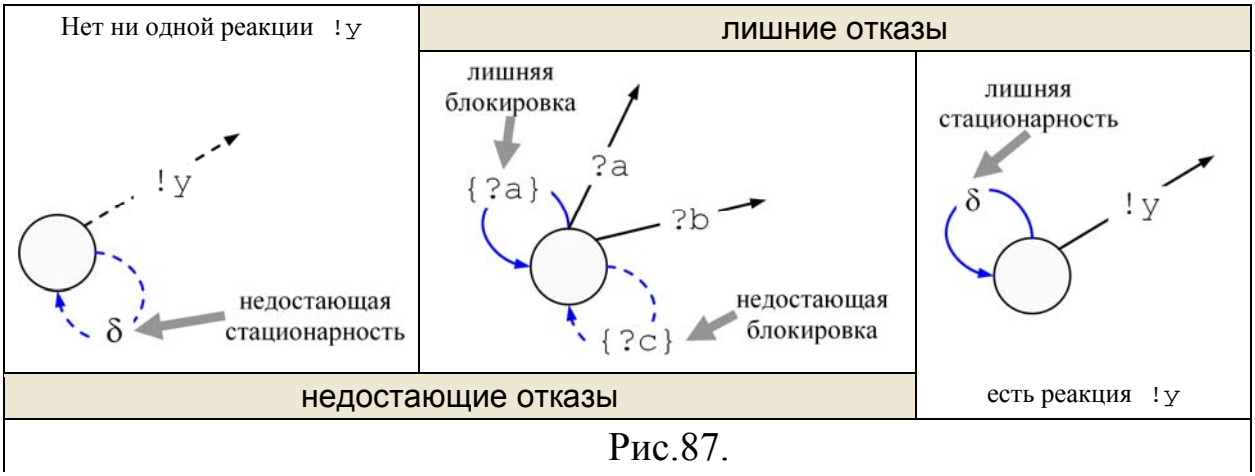


Рис.87.

2.3. Дерево $\mathbf{T}(O`)$ может содержать «лишние» отказы, из-за которых нарушается свойство (стаб.3). «Лишняя» трасса – это любая трасса, начинающаяся с «лишнего» отказа. (Смотри пример на Рис.87 для βγδ-модели, блокировок и стационарности.)

«Лишний» отказ – это такой отказ Q, что трассы отказов из $O`^*$ продолжаются как отказом $Q$, так и внешним действием из отказа $Q$. По свойству согласованности модели $\mathbf{\Sigma}$, такой «лишний» отказ $Q \notin O`$. Определим множество «лишних» трасс для «лишних» отказов:

$$\boldsymbol{Minus}(O`) = \{\ \langle Q\rangle\cdot\mu \mid\ Q\in\mathcal{P}(L)\setminus O`\ \&\ \mu\in L_{\mathcal{P}(L)\Delta\gamma}{}^*$$

$$\&\ \langle Q\rangle\in\mathbf{T}(O`)\ \&\ \exists z\in Q\ \langle z\rangle\in\mathbf{T}(O`)\ \}.$$

Удалим продолжение $O`^*$ «лишними» трассами:

$$\mathbf{T} = \mathbf{T}(O`) \setminus (O`^*\cdot\boldsymbol{Minus}(O`)).$$

2.3.1. Трасса $\rho\cdot\lambda$ не удаляется, так как λ начинается не с отказа, а ρ 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 $\rho\cdot\lambda\in\mathbf{T}$.

2.3.2. Множество $\mathbf{T}$ является деревом по построению.

2.3.3. Поскольку удаляемые трассы начинаются с отказа, остаётся верным $\mathbf{T}_c=(\mathbf{\Sigma}\ \textit{after}\ \rho)_c$.

2.3.4. Поскольку мы удаляем множество трасс вида $O`^*\cdot\ldots$, свойство (стаб.1) продолжает почти выполняться: $\forall\rho\in O`^*\ \mathbf{T}\ \textit{after}\ \rho = \mathbf{T}$. 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: $\forall\rho\in\mathcal{P}(L)^*\cap\mathbf{T}\ \mathbf{T}\ \textit{after}\ \rho = \mathbf{T}$.

2.3.5. Каждая удаляемая трасса имеет вид $\rho`\cdot\langle Q\rangle\cdot\ldots$, где $\rho`\in O`^*$, $Q\notin O`$ и $\exists z\in Q\ \langle z\rangle\in\mathbf{T}$. Иными словами, удаляется продолжение отказом, если есть продолжение внешними действиями из этого отказа. Поэтому свойство (стаб.2) сохраняется.

2.3.6. Наконец, теперь у нас нет «лишних» трасс, нарушающих свойство (стаб.3).

Итак, $\mathbf{T}$ – стабильное дерево, содержащее исходную трассу $\rho\cdot\lambda$ и $\mathbf{T}_c=(\mathbf{\Sigma}\ \textit{after}\ \rho)_c$.

3. Теперь покажем, что дерево $\mathbf{T}$ является ***F***-моделью.

По Лемме 2, $\mathbf{\Sigma}\ \textit{after}\ \rho$ является ***F***-моделью. По Лемме 3, $(\mathbf{\Sigma}\ \textit{after}\ \rho)_c$ является квази-моделью. По Лемме 1, $\mathbf{T}=O`^*\cdot\mathbf{T}_c$. Поскольку $\mathbf{T}$ стабильное дерево, для конкатенации $\mathbf{T}=O`^*\cdot\mathbf{T}_c$ выполнено условие $O` = \{\ Q\subseteq L\ |\ Q\cap\textit{head}(\mathbf{T}_c)=\varnothing\ \}$.

Поскольку $\mathbf{T}_c=(\mathbf{\Sigma}\ \textit{after}\ \rho)_c$, $\mathbf{T}_c$ является квази-моделью и не содержит трасс $\langle\Delta\rangle$ и $\langle\gamma\rangle$. Тем самым, выполнены условия Леммы 4 для $O`^*\cdot\mathbf{T}_c$ и $\mathbf{T}$ является ***F***-моделью.

4. Теперь мы можем построить для каждой трассы из $\mathbf{\Sigma}$, начинающейся с отказов, стабильную ***F***-модель, содержащую эту трассу. Множество таких ***F***-

моделей обозначим $\mathbf{\Sigma}_S$. Теперь, очевидно, $\mathbf{\Sigma}=\mathbf{\Sigma}_C\cup\cup\mathbf{\Sigma}_S$, что и требовалось доказать.

11. Доказательство Леммы 5

Утверждение непосредственно следует из определения операций ***drti***. Можно также заметить, что в доказательстве Теоремы 6 для каждой трассы $\rho\cdot\lambda\in\mathbf{\Sigma}$, где ρ – непустая трасса отказов, а трасса λ не начинается с отказа, то есть $\rho=\mathit{pref}(\rho\cdot\lambda)\neq\epsilon$, мы строили дерево $\mathbf{T}$ со свойствами:

$\mathit{drti}(\rho)=O`^*$,

$\mathbf{T}_C=(\mathbf{\Sigma}\ \mathit{after}\ \rho)_C$,

$O` = \{Q\subseteq L \mid Q\cap\mathit{head}(\mathbf{T}_C)=\varnothing\}$.

Напомним, что $\mathit{head}(\mathbf{T}_C)=L\cap\mathit{head}(\mathbf{T})$.

Также, для $Q\subseteq L$ верно $Q\cap\mathit{head}((\mathbf{\Sigma}\ \mathit{after}\ \rho)_C)=Q\cap\mathit{head}(\mathbf{\Sigma}\ \mathit{after}\ \rho)$.

Отсюда, для $\lambda=\epsilon$, имеем

$\mathit{drti}(\rho)\ =\{Q\subseteq L \mid Q\cap\mathit{head}(\mathbf{\Sigma}\ \mathit{after}\ \rho)=\varnothing\}^*\ =\mathcal{P}(L\setminus\mathit{head}(\mathbf{\Sigma}\ \mathit{after}\ \rho))^*$.

12. Доказательство Леммы 6

Допустим утверждение не верно. Тогда $\langle\gamma\rangle\notin\mathbf{\Sigma}$, $\langle\gamma\rangle\notin\mathbf{I}$ и

$\exists\sigma\in\mathit{SafeBy}(\mathbf{\Sigma})\cap\mathbf{I}\ \exists P\in\mathfrak{R}\cup\mathfrak{Q}$

P ***safe by*** $\mathbf{\Sigma}$ ***after*** σ & P ~~***safe in***~~ $\mathbf{I}$ ***after*** σ.

При условии $\langle\gamma\rangle\notin\mathbf{\Sigma}$ и $\langle\gamma\rangle\notin\mathbf{I}$ имеем $\epsilon\in\mathit{SafeBy}(\mathbf{\Sigma})\cap\mathit{SafeIn}(\mathbf{I})$.

Поэтому можно выбрать трассу σ такой, чтобы $\sigma=\mu\cdot\langle u\rangle$, где $\mu\in\mathit{SafeBy}(\mathbf{\Sigma})\cap\mathit{SafeIn}(\mathbf{I})$.

Тогда символ u ***safe by*** $\mathbf{\Sigma}$ ***after*** μ.

Поскольку $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$, должно быть u ***safe in*** $\mathbf{I}$ ***after*** μ.

А тогда $\sigma \in \mathit{SafeIn}(\mathbf{I})$, что противоречит допущению.

13. Доказательство Леммы 7

Если существует безопасная трасса спецификации $\mathbf{\Sigma}$, то $\langle\gamma\rangle \notin \mathbf{\Sigma}$, а тогда для безопасной реализации $\mathbf{I}$ должно быть тоже $\langle\gamma\rangle \notin \mathbf{I}$ и, следовательно, её пустая трасса безопасна.

Далее, по индукции, пусть непустая безопасная трасса спецификации принадлежит реализации $\sigma\cdot\langle u\rangle \in \mathit{SafeBy}(\mathbf{\Sigma}) \cap \mathbf{I}$, где $\sigma \in \mathit{SafeBy}(\mathbf{\Sigma}) \cap \mathit{SafeIn}(\mathbf{I})$ и для некоторого $P \in \mathfrak{R} \cup \mathfrak{Q}$ имеет место P ***safe by*** $\mathbf{\Sigma}$ ***after*** σ и $u \in P$ или $u = P$ и $P \in \mathfrak{R}$. Тогда, по гипотезе о безопасности, P ***safe in*** $\mathbf{I}$ ***after*** σ, что влечёт $\sigma\cdot\langle u\rangle \in \mathit{SafeIn}(\mathbf{I})$.

14. Доказательство Леммы 8

Допустим утверждение не верно. Тогда $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$ и

$\exists\sigma \in \mathit{SafeBy}(\mathbf{\Sigma}) \cap \mathbf{I}$ $\exists P$ ***safe by*** $\mathbf{\Sigma}$ ***after*** σ

$\mathit{obs}(\sigma, P, \mathbf{I}) \not\subseteq \mathit{obs}(\sigma, P, \mathbf{\Sigma})$.

Отсюда следует, что $\langle\gamma\rangle \notin \mathbf{\Sigma}$ и, поскольку $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$, имеем $\langle\gamma\rangle \notin \mathbf{I}$.

Значит $\epsilon \in \mathit{SafeBy}(\mathbf{\Sigma}) \cap \mathit{SafeIn}(\mathbf{I})$, и можно выбрать трассу σ такой, чтобы $\sigma = \mu\cdot\langle u\rangle$, где $\mu \in \mathit{SafeBy}(\mathbf{\Sigma}) \cap \mathit{SafeIn}(\mathbf{I})$.

Тогда символ u ***safe by*** $\mathbf{\Sigma}$ ***after*** μ.

Поскольку $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$, должно быть u ***safe in*** $\mathbf{I}$ ***after*** μ.

А тогда $\sigma \in \mathit{SafeIn}(\mathbf{I})$.

Поскольку $\mathbf{I}$ ***saco*** $\mathbf{\Sigma}$, должно быть $\mathit{obs}(\sigma, P, \mathbf{I}) \subseteq \mathit{obs}(\sigma, P, \mathbf{\Sigma})$, что противоречит допущению.

15. Доказательство Леммы 9

Утверждение непосредственно следует из определения тестовых и безопасных трасс. Пустая трасса является тестовой тогда и только тогда, когда она безопасна, то есть в спецификации нет γ-трассы. Непустая тестовая трасса σ либо является безопасной трассой $\sigma=\mu$, либо продолжает безопасную трассу μ безопасным символом u, а такое продолжение $\sigma=\mu\cdot\langle u\rangle$ является безопасной трассой тогда и только тогда, когда оно имеется в спецификации.

16. Доказательство Теоремы 7

Пусть в $\mathfrak{R}/\mathfrak{Q}$-семантике заданы ***F***-спецификация $\mathbf{\Sigma}\in\boldsymbol{FMODEL}(L)$ с отношением ***safe by***. Рассмотрим произвольную безопасную реализацию $\mathbf{I}$. Нам надо доказать, что строгий тестовый набор $\mathbf{TT}$, покрывающий все тестовые трассы спецификации, значимый и исчерпывающий.

1. Значимость. Нам надо показать, что $\mathbf{I}\ \boldsymbol{saco}\ \mathbf{\Sigma}\Rightarrow\mathbf{I}\ \boldsymbol{passes}\ \mathbf{TT}$. Это эквивалентно $\mathbf{I}\ \boldsymbol{saco}\ \mathbf{\Sigma}\Rightarrow\mathbf{I}\cap(\cup_{\circ}\boldsymbol{fail}(\mathbf{TT}))=\varnothing$, а это, в свою очередь, эквивалентно $\mathbf{I}\ \boldsymbol{\not saco}\ \mathbf{\Sigma}\Leftarrow\mathbf{I}\cap(\cup_{\circ}\boldsymbol{fail}(\mathbf{TT}))\neq\varnothing$. Пусть существует такой тест $\mathbf{T}\in\mathbf{TT}$ и такая его ***fail***-трасса $\sigma\in\boldsymbol{fail}(\mathbf{T})$, которая присутствует в реализации $\sigma\in\mathbf{I}$. Поскольку тест $\mathbf{T}$ строгий, $\sigma\in\boldsymbol{fail}(\mathbf{T})$ влечёт $\sigma\notin\mathbf{\Sigma}$. По определению тестовых трасс, существует такая безопасная трасса спецификации $\mu\in\boldsymbol{SafeBy}(\mathbf{\Sigma})$ и такой безопасный символ u ***safe by*** $\mathbf{\Sigma}$ ***after*** μ, что $\sigma=\mu\cdot\langle u\rangle$. Тогда $\mu\in\mathbf{I}\cap\boldsymbol{SafeBy}(\mathbf{\Sigma})$, и, по Лемме 7, $\mu\in\boldsymbol{SafeIn}(\mathbf{I})$. Имеем $\mu\in\boldsymbol{SafeBy}(\mathbf{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I})$, u ***safe by*** $\mathbf{\Sigma}$ ***after*** μ и $\mu\cdot\langle u\rangle\in\mathbf{I}$. Если бы реализация $\mathbf{I}$ была конформна, то было бы $\mu\cdot\langle u\rangle\in\mathbf{\Sigma}$, что не верно. Следовательно, $\mathbf{I}\ \boldsymbol{\not saco}\ \mathbf{\Sigma}$, что и требовалось доказать.

2. Исчерпываемость. Нам надо показать, что $\mathbf{I}\ \boldsymbol{saco}\ \mathbf{\Sigma}\Leftarrow\mathbf{I}\ \boldsymbol{passes}\ \mathbf{TT}$. Это эквивалентно $\mathbf{I}\ \boldsymbol{saco}\ \mathbf{\Sigma}\Leftarrow\mathbf{I}\cap(\cup_{\circ}\boldsymbol{fail}(\mathbf{TT}))=\varnothing$, а это, в свою очередь,

эквивалентно $\mathbf{I}$ ~~***saco***~~ $\mathbf{\Sigma} \Rightarrow \mathbf{I} \cap (\cup_{\circ} \boldsymbol{fail}(\mathbf{TT})) \neq \varnothing$. Поскольку реализация $\mathbf{I}$ безопасная, существует такая трасса $\mu \in \boldsymbol{SafeBy}(\mathbf{\Sigma}) \cap \boldsymbol{SafeIn}(\mathbf{I})$ и такой символ u ***safe by*** $\mathbf{\Sigma}$ ***after*** μ, что $\mu \cdot \langle u \rangle \in \mathbf{I}$, но $\mu \cdot \langle u \rangle \notin \mathbf{\Sigma}$. Трасса $\mu \cdot \langle u \rangle$ является тестовой трассой, а тестовый набор $\mathbf{TT}$ покрывает все тестовые трассы. Поэтому существует такой тест $\mathbf{T} \in \mathbf{TT}$, который содержит эту трассу $\mu \cdot \langle u \rangle \in \mathbf{T}$. Поскольку тест $\mathbf{T}$ строгий, а $\mu \cdot \langle u \rangle \notin \mathbf{\Sigma}$, то трасса $\mu \cdot \langle u \rangle \in \boldsymbol{fail}(\mathbf{T})$. Имеем $\mu \cdot \langle u \rangle \in \mathbf{I} \cap \boldsymbol{fail}(\mathbf{T})$, следовательно, $\mathbf{I} \cap \boldsymbol{fail}(\mathbf{T}) \neq \varnothing$ и $\mathbf{I} \cap (\cup_{\circ} \boldsymbol{fail}(\mathbf{TT})) \neq \varnothing$, что и требовалось доказать.

17. Доказательство Теоремы 8

1. Существование примитивного теста для каждой тестовой трассы σ непосредственно следует из определения примитивного теста и тестовой трассы. Действительно, для каждого префикса μ, продолжаемого в трассе символом u, то есть $\mu \cdot \langle u \rangle \leq \sigma$, всегда можно выбрать кнопку “P”, безопасную в спецификации после μ и такую, что $u \in P$ или $u = P$ (для $P \in \mathfrak{R}$). После этого для каждого $t \neq u$ такого, что $t \in P$ или $t = P$ (для $P \in \mathfrak{R}$), берём трассу $\mu \cdot \langle t \rangle$ как максимальную трассу теста. Трассу σ также выбираем как максимальную трассу теста (если она не пуста, она совпадает с одной из трасс вида $\mu \cdot \langle t \rangle$). Добавляем в тест все префиксы максимальных трасс. Устанавливаем для каждой максимальной трассы соответствующий вердикт, определяемый условием строгого теста: ***pass***, если трасса принадлежит спецификации, и ***fail*** в противном случае. Очевидно, мы получим ограниченный управляемый строгий тест.

2. Поскольку примитивный тест можно построить для каждой тестовой трассы спецификации и он содержит саму эту трассу, множество примитивных

тестов покрывает все тестовые трассы спецификации. Учитывая также, что примитивные тесты – это строгие тесты, по Теореме 7 получаем, что множество примитивных тестов является полным набором тестов.

18. Доказательство Теоремы 9

Множество ***F***-трасс LTS $\mathbf{s}$ – это множество трасс LTS $L2L_F(\mathbf{s})$. Оно является деревом, поскольку деревом является множество маршрутов LTS, а префикс трассы маршрута является трассой префикса маршрута.

1. Допустимость непосредственно следует из способа построения ***F***-трасс LTS: переход по разрушению или дивергенции ведёт в терминальное $\mathbf{t}$-состояние.

2. Согласованность непосредственно следует из способа построения ***F***-трасс LTS: петля отказа P определяется в стабильном состоянии (из него не ведут τ- и γ-переходы) при отсутствии переходов из этого состояния по внешним действиям из P.

3. $\mathcal{P}(L)$-конвергентность. Если ***F***-трасса не содержит дивергенции и разрушения, она не заканчивается в состоянии $\mathbf{t}$, то есть заканчивается в некотором множестве состояний LTS $\mathbf{s}$. Если ***F***-трасса заканчивается только в нестабильных состояниях, то из каждого такого состояния выходит γ- или τ-переход. Следовательно, хотя бы в одном состоянии должен быть γ-переход или начинаться бесконечный маршрут τ-переходов (состояние дивергентно). В этом случае ***F***-трасса продолжается разрушением и/или дивергенцией. Поэтому, если ***F***-трасса не содержит и не продолжается дивергенцией и разрушением, она заканчивается хотя бы в одном стабильном состоянии s. А тогда для каждого отказа P ***F***-трасса продолжается либо отказом P, если в состоянии s нет переходов по действиям из P, либо действием из этого отказа.

4. Замкнутость по ***d***-операции также непосредственно следует из способа построения ***F***-трасс LTS: отказы моделируются петлями в стабильных состояниях.
5. $\mathcal{P}(\mathtt{L})$-полнота (замкнутость по ***i***-операции). Пусть ***F***-трасса σ заканчивается отказом и не продолжается внешними действиями из некоторого отказа $\mathtt{P}$. Такая трасса в LTS заканчивается в стабильных состояниях, в каждом из которых нет переходов по внешним действиям из $\mathtt{P}$. Следовательно, в каждом из этих состояний определена петля по отказу $\mathtt{P}$. Если какая-то трасса λ продолжает трассу σ, то она начинается в одном из этих состояний. Следовательно, при проходе через такое состояние возможна трасса $\sigma\cdot\langle\mathtt{P}\rangle\cdot\lambda$.

19. Доказательство Теоремы 10

Утверждение непосредственно следует из определения объединения LTS.

20. Доказательство Теоремы 11

Утверждение непосредственно следует из определения правил работы LTS и машины тестирования.

21. Доказательство Леммы 10

Состояние, соответствующее ***F***-трассе $\mu\cdot\langle\gamma\rangle\in\boldsymbol{\Sigma}$, стабильно, поскольку в нём не определяются никакие переходы (по допустимости ***F***-модели, ***F***-трасса, заканчивающаяся на разрушение, ничем не продолжается). Имеем $\boldsymbol{init}(\mu\cdot\langle\gamma\rangle)=\varnothing=\varnothing\cap\mathtt{L}=\boldsymbol{head}(\{\epsilon\})\cap\mathtt{L}=\boldsymbol{head}(\boldsymbol{\Sigma}\ \boldsymbol{after}\ \mu\cdot\langle\gamma\rangle)\cap\mathtt{L}$.

Состояние, соответствующее ***F***-трассе $\mu\cdot\mathbf{o}\in\boldsymbol{\Sigma}$, где $\mathbf{o}\in\mathcal{P}(\mathtt{L})^*$ и $\mathbf{o}\neq\epsilon$, стабильно: из него нет τ-перехода по правилам вывода, и из него нет γ-перехода, поскольку после отказа в ***F***-трассе не может быть разрушения по

согласованности ***F***-модели. В этом состоянии определён переход по внешнему действию $z \in L$ тогда и только тогда, когда $\mu \cdot o \cdot \langle z \rangle \in \Sigma$, что эквивалентно $z \in \mathit{head}(\Sigma \ \mathit{after} \ \mu \cdot o)$.

Тем самым, $\mathit{init}(\mu \cdot o) = \mathit{head}(\Sigma \ \mathit{after} \ \mu \cdot o) \cap L$.

Утверждение о конвергентности состояний: τ-переход $\mu \text{—}\tau\rightarrow \mu \cdot o$, определяемый правилом вывода (2), ведёт в стабильное состояние, в котором нет τ-переходов, а τ-переход $\mu \text{—}\tau\rightarrow \mu$ определяется правилом вывода (3) тогда и только тогда, когда $\mu \cdot \langle \Delta \rangle \in \Sigma$.

Нам осталось показать, что достижимое состояние, соответствующее ***F***-трассе $\mu \in \Sigma$, не заканчивающейся разрушением или отказом, нестабильно.

По построению, ***F***-трасса, соответствующая достижимому состоянию, не заканчивается дивергенцией.

Если ***F***-трасса μ продолжается в Σ γ-действием, то, по правилу вывода (1), в состоянии, соответствующем ***F***-трассе μ, определяется γ-переход и оно нестабильно. Если ***F***-трасса μ продолжается в Σ Δ-действием, то, по правилу вывода (3), в состоянии, соответствующем ***F***-трассе μ, определяется τ-петля и оно нестабильно.

По $\mathcal{P}(L)$-конвергентности, ***F***-трасса μ, не заканчивающаяся и не продолжающаяся в Σ дивергенцией и разрушением, продолжается в Σ пустым отказом. А тогда, по правилу вывода (2), в состоянии, соответствующем ***F***-трассе μ, определяется τ-переход и оно нестабильно.

22. Доказательство Леммы 11

По Лемме 10, маршрут S заканчивается в дивергентном состоянии тогда и только тогда, когда $\langle \omega(S) \cdot \Delta \rangle \in \Sigma$. Поэтому нам достаточно доказать, что

$\boldsymbol{F}(S)\setminus\{\langle\omega(S)\cdot\Delta\rangle\}=\boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu)$. Доказательство будем вести индукцией по длине маршрута S.

Пустой маршрут начинается и заканчивается в начальном состоянии, соответствующем пустой трассе $\omega((\epsilon,\epsilon))=\epsilon$. По Лемме 10, начальное состояние нестабильно. Поэтому $\boldsymbol{F}(\epsilon)\setminus\{\langle\Delta\rangle\}=\{\epsilon\}=\boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\epsilon)$.

Пусть утверждение верно для маршрута S, заканчивающегося в состоянии $\omega(S)=\mu\in\Sigma$: $\boldsymbol{F}(S)\setminus\{\mu\cdot\langle\Delta\rangle\}=\boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu)$.

Рассмотрим возможные маршруты $S1$, каждый из которых является продолжением маршрута S одним переходом.

1. Переход по правилу вывода (1): $\mu\text{—}z\rightarrow\mu\cdot\langle z\rangle$, $z\in L_\gamma$, $\mu\cdot\langle z\rangle\in\Sigma$.

Имеем: $\omega(S1)=\mu\cdot\langle z\rangle$. По Лемме 10, состояние, соответствующее трассе $\mu\cdot\langle z\rangle$, нестабильно. Поэтому:

$\boldsymbol{F}(S1)\setminus\{\langle\mu\cdot\langle z\rangle\cdot\Delta\rangle\}$

$=\ (\boldsymbol{F}(S)\setminus\{\mu\cdot\langle\Delta\rangle\})\cdot\{\langle z\rangle\}$

$=$ (* по предположению шага индукции *)

$=\ \boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu)\cdot\{\langle z\rangle\}$

$=\ \boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu\cdot\langle z\rangle)$.

2. Переход по правилу вывода (2): $\mu\text{—}\tau\rightarrow\mu\cdot o$, $o\in\mathcal{P}(L)^*$, $o\neq\epsilon$, $\mu\cdot o\in\Sigma$.

Имеем $\omega(S1)=\mu\cdot o$. По Лемме 10, состояние, соответствующее трассе $\mu\cdot o$, стабильно и $\boldsymbol{init}(\mu\cdot o)\ =\ \boldsymbol{head}(\Sigma\ \boldsymbol{after}\ \mu\cdot o)\cap L$.

По определению множества отказов, порождаемых стабильным состоянием, $\boldsymbol{f}(\mu\cdot o)=\mathcal{P}(L\setminus\boldsymbol{init}(\mu\cdot o))$.

Поэтому

$\boldsymbol{f}(\mu\cdot o) = \mathcal{P}(L\setminus(\boldsymbol{head}(\Sigma\ \boldsymbol{after}\ \mu\cdot o)\cap L))$

$= \mathcal{P}(L\setminus\boldsymbol{head}(\Sigma\ \boldsymbol{after}\ \mu\cdot o)) = \mathcal{P}(L\setminus\boldsymbol{head}(\Sigma\ \boldsymbol{after}\ \mu\ \boldsymbol{after}\ o))$.

По Лемме 5,

$\boldsymbol{drti}_{\mathrm{in}\,\Sigma\ \boldsymbol{after}\ \mu}(o) = \mathcal{P}(L\setminus\boldsymbol{head}(\Sigma\ \boldsymbol{after}\ \mu\ \boldsymbol{after}\ o))^*$.

Поэтому:

$\boldsymbol{F}(S1)\setminus\{\mu\cdot o\cdot\langle\Delta\rangle\}$

$= (\boldsymbol{F}(S)\setminus\{\mu\cdot\langle\Delta\rangle\})\cdot\boldsymbol{f}(\mu\cdot o)^*$

$= (\boldsymbol{F}(S)\setminus\{\mu\cdot\langle\Delta\rangle\})\cdot\boldsymbol{drti}_{\mathrm{in}\,\Sigma\ \boldsymbol{after}\ \mu}(o)$

= (* по предположению шага индукции *)

$= \boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu)\cdot\boldsymbol{drti}_{\mathrm{in}\,\Sigma\ \boldsymbol{after}\ \mu}(o)$

= (* поскольку трасса μ не заканчивается на отказ *)

$= \boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu\cdot o)$.

3. Переход по правилу вывода (3): $\mu\text{—}\tau\rightarrow\mu$.

Имеем $\omega(S1)=\mu$ и

$\boldsymbol{F}(S1)\setminus\{\mu\cdot\langle\Delta\rangle\}$

$= \boldsymbol{F}(S)\setminus\{\mu\cdot\langle\Delta\rangle\}$

= (* по предположению шага индукции *)

$= \boldsymbol{drti}_{\mathrm{in}\,\Sigma}(\mu)$.

23. Доказательство Теоремы 12

1. Докажем вложенность $\Sigma\subseteq\boldsymbol{F}(\boldsymbol{s})$.

Нам достаточно показать, что для каждой трассы $\mu\in\Sigma$ имеет место $(\boldsymbol{s}\ \boldsymbol{after}\ \mu)\neq\varnothing$. Мы будем доказывать более сильное утверждение:

$\mu\in(\mathbf{S}\ \textit{after}\ \mu)$, если μ не заканчивается Δ, и $\mu\in(\mathbf{S}\ \textit{after}\ \mu\cdot\langle\Delta\rangle)$, если $\mu\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$.

Доказательство будем вести индукцией по длине трассы $\mu\in\mathbf{\Sigma}$.

Поскольку начальное состояние LTS $\mathbf{S}$ соответствует пустой трассе, $\epsilon\in(\mathbf{S}\ \textit{after}\ \epsilon)$.

Пусть утверждение верно для трассы $\mu\in\mathbf{\Sigma}$, не заканчивающейся отказом. Рассмотрим возможные минимальные продолжения трассы, также не заканчивающиеся отказом.

1.1. $\mu\cdot\langle z\rangle\in\mathbf{\Sigma}$, где $z\in L_\gamma$.

По правилу вывода (1), имеется переход $\mu\text{—}z\rightarrow\mu\cdot\langle z\rangle$.

Поэтому $\mu\in(\mathbf{S}\ \textit{after}\ \mu)$ влечёт $\mu\cdot\langle z\rangle\in(\mathbf{S}\ \textit{after}\ \mu\cdot\langle z\rangle)$.

1.2. $\mu\cdot\mathbf{o}\in\mathbf{\Sigma}$, где $\mathbf{o}\in\mathcal{P}(L)^*$ и $\mathbf{o}\neq\epsilon$.

По правилу вывода (2), имеется переход $\mu\text{—}\tau\rightarrow\mu\cdot\mathbf{o}$.

По согласованности трассовой модели,

$\textit{Im}(\mathbf{o})\ \subseteq\ \mathcal{P}(L\setminus\textit{head}(\mathbf{\Sigma}\ \textit{after}\ \mu\cdot\mathbf{o}))$.

По Лемме 10, $\textit{init}(\mu\cdot\mathbf{o})=\textit{head}(\mathbf{\Sigma}\ \textit{after}\ \mu\cdot\mathbf{o})\cap L$.

По определению множества отказов, порождаемых стабильным состоянием, $\boldsymbol{f}(\mu\cdot\mathbf{o})=\mathcal{P}(L\setminus\textit{init}(\mu\cdot\mathbf{o}))$.

Поэтому

$\boldsymbol{f}(\mu\cdot\mathbf{o})=\mathcal{P}(L\setminus(\textit{head}(\mathbf{\Sigma}\ \textit{after}\ \mu\cdot\mathbf{o})\cap L))=\mathcal{P}(L\setminus\textit{head}(\mathbf{\Sigma}\ \textit{after}\ \mu\cdot\mathbf{o}))$.

Таким образом, $\textit{Im}(\mathbf{o})\subseteq\boldsymbol{f}(\mu\cdot\mathbf{o})$.

Поэтому $\mu\in(\mathbf{S}\ \textit{after}\ \mu)$ влечёт $\mu\cdot\mathbf{o}\in(\mathbf{S}\ \textit{after}\ \mu\cdot\mathbf{o})$.

1.3. $\mu\cdot\mathbf{o}\cdot\langle z\rangle\in\mathbf{\Sigma}$, где $\mathbf{o}\in\mathcal{P}(L)^*$, $\mathbf{o}\neq\epsilon$ и $z\in L$

Из доказанного случая для трассы $\mu\cdot\mathbf{o}$ следует $\mu\cdot\mathbf{o}\in(\mathbf{s}\ \textit{after}\ \mu\cdot\mathbf{o})$.

По правилу вывода (1), имеется переход $\mu\cdot\mathbf{o}\text{—}z\rightarrow\mu\cdot\mathbf{o}\cdot\langle z\rangle$.

Поэтому $\mu\cdot\mathbf{o}\cdot\langle z\rangle\in(\mathbf{s}\ \textit{after}\ \mu\cdot\mathbf{o}\cdot\langle z\rangle)$.

1.4. $\mu\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$.

По Лемме 10, состояние μ дивергентно.

Поэтому $\mu\in(\mathbf{s}\ \textit{after}\ \mu\cdot\langle\Delta\rangle)$.

Заметим, что трассы $\mu\cdot\mathbf{o}\cdot\langle\gamma\rangle$ не может быть по согласованности ***F***-модели.

2. Теперь докажем вложенность $\mathbf{\Sigma}\supseteq\boldsymbol{F}(\mathbf{s})$.

Рассмотрим маршрут $S\in\boldsymbol{R}(\mathbf{s})$.

По Лемме 11, $\boldsymbol{F}(S)=\textit{drti}_{\text{in}\,\mathbf{\Sigma}}\circ\omega(S)\cup(\{\omega(S)\cdot\langle\Delta\rangle\}\cap\mathbf{\Sigma})$.

Очевидно, $\{\omega(S)\cdot\langle\Delta\rangle\}\cap\mathbf{\Sigma}\subseteq\mathbf{\Sigma}$.

В силу замкнутости и $\mathcal{P}(L)$-полноты ***F***-модели, $\textit{drti}_{\text{in}\,\mathbf{\Sigma}}\circ\omega(S)\subseteq\mathbf{\Sigma}$.

Тем самым, $\boldsymbol{F}(S)\subseteq\mathbf{\Sigma}$.

Поскольку множество ***F***-трасс LTS совпадает с объединением множеств ***F***-трасс её маршрутов $\boldsymbol{F}(\mathbf{s})=\cup\circ\boldsymbol{F}\circ\boldsymbol{R}(\mathbf{s})$, имеем $\boldsymbol{F}(\mathbf{s})\subseteq\mathbf{\Sigma}$, что и требовалось доказать.

Совокупность утверждений **1.** и **2.** доказывает общее утверждение.

24. Доказательство Леммы 12

Дерево $\{\epsilon\}$ не является ***F***-моделью (нет $\mathcal{P}(L)$-конвергентности). Поэтому, по правилам вывода, в состоянии, соответствующем дереву $\{\epsilon\}$, не определяются никакие переходы. Имеем $\textit{init}(\{\epsilon\})=\varnothing=\textit{head}(\{\epsilon\})\cap L$.

Стабильное дерево не содержит γ- и Δ-трасс, поэтому для него не применимы правила вывода (1) для $z=\gamma$ и правило вывода (3). Также для стабильного дерева не применимо правило вывода (2). Поэтому в состоянии, соответствующем стабильному дереву $\mathbf{T}$, не определены τ- и γ-переходы. По правилу вывода (1), в этом состоянии определён переход по внешнему действию $z\in L$ тогда и только тогда, когда $z\in \boldsymbol{head}(\mathbf{T})$. Тем самым, это состояние стабильно и $\boldsymbol{init}(\mathbf{T})=\boldsymbol{head}(\mathbf{T})\cap L$.

Утверждение о конвергентности состояний: τ-переход $\mathbf{T}\text{—}\tau\rightarrow\mathbf{T}`$, определяемый правилом вывода (2), ведёт в стабильное состояние, в котором нет τ-переходов, а τ-переход $\mathbf{T}\text{—}\tau\rightarrow\mathbf{T}$ определяется правилом вывода (3) тогда и только тогда, когда $\langle\Delta\rangle\in\mathbf{T}$. Поэтому состояние конвергентно тогда и только тогда, когда оно соответствует дереву, в котором нет трассы $\langle\Delta\rangle$.

Нам осталось показать, что состояние, соответствующее нестабильному дереву $\mathbf{T}$, нестабильно.

Если $\mathbf{T}$ содержит γ-трассу, то, по правилу вывода (1), в состоянии $\mathbf{T}$ определяется γ-переход, и оно нестабильно. Если $\mathbf{T}$ содержит Δ-трассу, то, по правилу вывода (3), в состоянии $\mathbf{T}$ определяется τ-петля, и оно нестабильно.

Пусть $\mathbf{T}$ не содержит Δ- и γ-трасс. Поскольку дерево $\mathbf{T}$ является $\boldsymbol{F}$-моделью, по $\mathcal{P}(L)$-конвергентности, его пустая трасса продолжается в нём пустым отказом. Следовательно, в разложении $\mathbf{T}=\mathbf{T}_c\cup\cup\mathbf{T}_s$ множество стабильных деревьев $\mathbf{T}_s$ не пусто. Поэтому, по правилу вывода (2), в состоянии $\mathbf{T}$ определяется τ-переход, и оно нестабильно.

25. Доказательство Теоремы 13

Мы будем ниже рассматривать только достижимые состояния LTS.

1. Докажем вложенность $\boldsymbol{\Sigma}\subseteq\boldsymbol{F}(\mathbf{s})$: $\forall\mu\in\boldsymbol{\Sigma}\ \ \mu\in\boldsymbol{F}(\mathbf{s})$.

1.1. Сначала докажем утверждение 1 для ***F***-трассы μ, которая не заканчивается отказом.

Мы будем доказывать более сильный вариант утверждения 1.1:

для каждой ***F***-трассы $\mu\cdot\lambda\in\mathbf{\Sigma}$, где μ не заканчивается отказом, имеет место $\mu\in\boldsymbol{F}(\mathbf{s})$ и, если μ не заканчивается Δ- или γ-действием, то существует такое состояние $\mathbf{T}\in(\mathbf{s}\ \textit{after}\ \mu)$, что $\lambda\in\mathbf{T}$.

Доказательство будем вести индукцией по числу базовых действий в ***F***-трассе μ, то есть по длине проекции $\mu^{\downarrow}\mathtt{L}_{\Delta\gamma}$.

Для $\mu=\epsilon$ и любой ***F***-трассы $\epsilon\cdot\lambda\in\mathbf{\Sigma}$ пустая трасса $\epsilon\in\boldsymbol{F}(\mathbf{s})$ и заканчивается в начальном состоянии $\mathbf{\Sigma}\in(\mathbf{s}\ \textit{after}\ \epsilon)$, а $\lambda\in\mathbf{\Sigma}$.

Пусть утверждение 1.1 верно для ***F***-трассы μ и докажем его для трассы $\mu\cdot\mathbf{o}\cdot\langle z\rangle\in\mathbf{\Sigma}$, где $\mathbf{o}\in\mathcal{P}(\mathtt{L})^{*}$ трасса отказов (быть может, пустая), а $z\in\mathtt{L}_{\Delta\gamma}$.

По допустимости ***F***-модели, μ не заканчивается Δ- или γ-действием.

Пусть $\mu\cdot\mathbf{o}\cdot\langle z\rangle\cdot\lambda\in\mathbf{\Sigma}$.

По предположению индукции, $\mu\in\boldsymbol{F}(\mathbf{s})$ и существует такое состояние $\mathbf{T}\in(\mathbf{s}\ \textit{after}\ \mu)$, что $\mathbf{o}\cdot\langle z\rangle\cdot\lambda\in\mathbf{T}$.

1.1.1. Пусть $\mathbf{o}=\epsilon$.

Если $z\in\mathtt{L}_{\gamma}$, то, по правилу вывода (1), трасса $\mu\cdot\langle z\rangle\in\boldsymbol{F}(\mathbf{s})$.

Если также $z\neq\gamma$, то

$(\mathbf{T}\ \textit{after}\ \langle z\rangle)\in(\mathbf{s}\ \textit{after}\ \mu\cdot\langle z\rangle)$ и $\lambda\in(\mathbf{T}\ \textit{after}\ \langle z\rangle)$.

Если $z=\Delta$, то, по правилу вывода (3), трасса $\mu\cdot\langle\Delta\rangle\in\boldsymbol{F}(\mathbf{s})$.

1.1.2. Пусть $\mathbf{o}\neq\epsilon$.

По согласованности ***F***-модели, $z\neq\gamma$ и $z\neq\Delta$.

Если дерево $\mathbf{T}$ стабильно, обозначим $\mathbf{T`}=\mathbf{T}$ (тогда $\mathbf{T}\Rightarrow\mathbf{T`}$). В противном случае, по правилу вывода (2), для некоторого $\mathbf{T`}\in\mathbf{T}_s$ выполняется $\mathbf{o}\cdot\langle z\rangle\cdot\lambda\in\mathbf{T`}$, и $\mathbf{T}\text{—}\tau\rightarrow\mathbf{T`}$ (также $\mathbf{T}\Rightarrow\mathbf{T`}$).

По Лемме 12, $\mathit{init}(\mathbf{T`})=\mathit{head}(\mathbf{T`})\cap L$, что влечёт $\mathbf{o}\in\boldsymbol{F}(\mathbf{T`})$, где $\boldsymbol{F}(\mathbf{T`})$ множество $\boldsymbol{F}$-трасс LTS $\mathbf{s}$ в состоянии $\mathbf{T`}$.

А тогда $\mu\in\boldsymbol{F}(\mathbf{s})$ & $\mathbf{T}\in(\mathbf{s}\ \mathit{after}\ \mu)$ & $\mathbf{T}\Rightarrow\mathbf{T`}$ & $\mathbf{o}\in\boldsymbol{F}(\mathbf{T`})$ влечёт $\mu\cdot\mathbf{o}\in\boldsymbol{F}(\mathbf{s})$ и $\mathbf{T`}\in(\mathbf{s}\ \mathit{after}\ \mu\cdot\mathbf{o})$.

Из $\mathbf{T`}$ ***stable*** и $\mathbf{o}\cdot\langle z\rangle\cdot\lambda\in\mathbf{T`}$ следует $\langle z\rangle\cdot\lambda\in\mathbf{T`}$ (правило `стаб.1`).

Отсюда, по правилу вывода (1), $\mu\cdot\mathbf{o}\cdot\langle z\rangle\in\boldsymbol{F}(\mathbf{s})$, $(\mathbf{T`}\ \mathit{after}\ \langle z\rangle)\in(\mathbf{s}\ \mathit{after}\ \mu\cdot\mathbf{o}\cdot\langle z\rangle)$ и $\lambda\in(\mathbf{T`}\ \mathit{after}\ \langle z\rangle)$.

Утверждение 1.1 доказано.

1.2. Теперь докажем первое утверждение для трассы, заканчивающейся отказом.

Такая трасса имеет вид $\mu\cdot\mathbf{o}\in\Sigma$, где μ не заканчивается отказом, а $\mathbf{o}\in\mathcal{P}(L)^*$ непустая трасса отказов $\mathbf{o}\neq\epsilon$. По допустимости $\boldsymbol{F}$-модели, μ не заканчивается символами γ или Δ.

Тогда, по утверждению 1.1, $\mu\in\boldsymbol{F}(\mathbf{s})$ и существует такое состояние $\mathbf{T}\in(\mathbf{s}\ \mathit{after}\ \mu)$, что $\mathbf{o}\in\mathbf{T}$.

Если дерево $\mathbf{T}$ стабильно, обозначим $\mathbf{T`}=\mathbf{T}$ (тогда $\mathbf{T}\Rightarrow\mathbf{T`}$). В противном случае, по правилу вывода (2), для некоторого $\mathbf{T`}\in\mathbf{T}_s$ выполняется $\mathbf{o}\in\mathbf{T`}$, и $\mathbf{T}\text{—}\tau\rightarrow\mathbf{T`}$ (также $\mathbf{T}\Rightarrow\mathbf{T`}$).

По Лемме 12, $\mathit{init}(\mathbf{T`})=\mathit{head}(\mathbf{T`})\cap L$, что влечёт $\mathbf{o}\in\boldsymbol{F}(\mathbf{T`})$, где $\boldsymbol{F}(\mathbf{T`})$ множество $\boldsymbol{F}$-трасс LTS $\mathbf{s}$ в состоянии $\mathbf{T`}$.

А тогда $\mu\in \boldsymbol{F}(\mathbf{s})$ & $\mathbf{T}\in(\mathbf{s}\ \textit{after}\ \mu)$ & $\mathbf{T}\Rightarrow\mathbf{T}'$ & $\mathbf{o}\in \boldsymbol{F}(\mathbf{T}')$

влечёт $\mu\cdot\mathbf{o}\in \boldsymbol{F}(\mathbf{s})$.

Первое утверждение доказано.

2. Теперь докажем обратную вложенность $\boldsymbol{\Sigma}\supseteq \boldsymbol{F}(\mathbf{s})$.

Поскольку множество ***F***-трасс LTS $\mathbf{s}$ совпадает с множеством трасс её ***F***-маршрутов, нам достаточно показать, что для каждого ***F***-маршрута S его трасса $\sigma=\textit{trace}(S)\in\boldsymbol{\Sigma}$. Мы будем доказывать также, что, если ***F***-маршрут не заканчивается в $\mathbf{t}$-состоянии (после Δ- или γ-перехода), то $\omega(S)=\mathbf{T}\subseteq(\boldsymbol{\Sigma}\ \textit{after}\ \sigma)$.

Доказательство будем вести индукцией по длине ***F***-маршрута S.

Очевидно, что для пустого ***F***-маршрута второе утверждение верно:

$\textit{trace}((\boldsymbol{\Sigma},\epsilon))=\epsilon\in\boldsymbol{\Sigma}$ и $\omega((\boldsymbol{\Sigma},\epsilon))=\boldsymbol{\Sigma}\subseteq(\boldsymbol{\Sigma}\ \textit{after}\ \epsilon)$.

Пусть утверждение верно для ***F***-маршрута S и докажем его для ***F***-маршрута $S'=S\cdot\langle e\rangle$, где e один переход. По определению ***F***-маршрута, S не содержит Δ- или γ-перехода.

По предположению индукции,

$\sigma=\textit{trace}(S)\in\boldsymbol{\Sigma}$ и $\omega(S)=\mathbf{T}\subseteq(\boldsymbol{\Sigma}\ \textit{after}\ \sigma)$.

Переход e порождён одним из правил вывода, либо является Δ- или γ-переходом, либо петлёй отказа.

2.1. Для правила вывода (1) и $z\neq\gamma$: $\textit{trace}(S')=\sigma\cdot\langle z\rangle$, $\sigma\cdot\langle z\rangle\in\boldsymbol{\Sigma}$ и

$\omega(S')=(\mathbf{T}\ \textit{after}\ \langle z\rangle)\subseteq(\boldsymbol{\Sigma}\ \textit{after}\ \sigma\cdot\langle z\rangle)$.

2.2. Для правила вывода (1) и $z=\gamma$: $\textit{trace}(S')=\sigma\cdot\langle z\rangle$, $\sigma\cdot\langle z\rangle\in\boldsymbol{\Sigma}$.

2.3. Для правила вывода (2): $\textit{trace}(S')=\sigma$, $\sigma\in\boldsymbol{\Sigma}$ и

$\omega(S')=\mathbf{T}'\subseteq(\boldsymbol{\Sigma}\ \textit{after}\ \sigma)$.

2.4. Для петли по отказу $\circ$: $\mathit{trace}(S`)=\sigma\cdot\langle\circ\rangle$ и

$\omega(S`)=\mathbf{T}_{\subseteq}(\mathbf{\Sigma}\ \mathit{after}\ \sigma)$.

Заметим, что S не содержит Δ- или γ-переходов.

По Лемме 12, дерево $\mathbf{T}$ стабильно и $\mathit{init}(\mathbf{T})=\mathit{head}(\mathbf{T})\cap L$.

По конвергентности стабильного дерева, $\langle\circ\rangle\in\mathbf{T}$.

Поэтому $\sigma\cdot\langle\circ\rangle\in\mathbf{\Sigma}$.

По свойству замкнутости ***F***-модели,

$\mathbf{T}=(\mathbf{T}\ \mathit{after}\ \langle\circ\rangle)\subseteq(\mathbf{\Sigma}\ \mathit{after}\ \sigma\cdot\langle\circ\rangle)$.

2.5. Для правила вывода (3): $\mathit{trace}(S`)=\sigma\cdot\langle\Delta\rangle$, $\sigma\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$.

Второе утверждение доказано.

Учитывая первое утверждение, общее утверждение доказано.

26. Доказательство Теоремы 14

Непосредственно следует из Теоремы 9 и Теоремы 12 (или Теоремы 13).

27. Доказательство Теоремы 15

Оба утверждения непосредственно следуют из определения преобразований ***Tt2Lt*** и $\underline{L}_\theta 2L_{\mathcal{P}(L)}$.

1. Нам нужно показать, что для каждого трассового теста $\mathbf{T}\in\mathit{Ttests}(F(\mathbf{s}),\mathfrak{R},\mathfrak{Q})$ имеет место $\mathit{traces}\circ L_\theta 2L_{\beta\delta}\circ\mathit{Tt2Lt}(\mathbf{T})=\mathbf{T}$. А это непосредственно следует из определения преобразований ***Tt2Lt*** и $\underline{L}_\theta 2L_{\mathcal{P}(L)}$.
2. Согласованность вердиктов также непосредственно следует из определения преобразований ***Tt2Lt*** и $\underline{L}_\theta 2L_{\mathcal{P}(L)}$.

28. Доказательство Теоремы 16

Обозначим $\mathbf{T}=\boldsymbol{Tt2Lt}(\mathrm{T})$.

Пусть в композиции $\mathbf{I}\,|\!|\,\mathbf{T}$ состояние $\mathtt{it}$ достижимо: $\mathtt{it}\in\boldsymbol{der}(\mathtt{i}_0\mathtt{t}_0)$.

Это может быть тогда и только тогда, когда существует такая $\mathfrak{R}$-трасса реализации $\sigma\in\boldsymbol{F}(\mathbf{I})$, что в преобразованной тестовой LTS $\underline{\boldsymbol{L}}_0\boldsymbol{2L}_{\mathcal{P}(\mathrm{L})}(\mathbf{T})$ есть такая же трасса $\sigma\in\boldsymbol{traces}\circ\underline{\boldsymbol{L}}_0\boldsymbol{2L}_{\mathcal{P}(\mathrm{L})}(\mathbf{T})$, заканчивающаяся в состоянии $\mathtt{t}$. Последнее, по Теореме 15, эквивалентно $\sigma\in\mathbf{T}$.

Отсюда, по Теореме 15 в части согласованности вердиктов, а также по определению ***passes*** для трассовых моделей и LTS-моделей, непосредственно следует доказываемое утверждение.

29. Доказательство Леммы 13

1. Необходимость. Нам достаточно показать, что, если состояние $\mathtt{s}$ Б-достижимо, но не БЛК-ветвящееся или не тау-ограниченное, то найдётся безопасная $\mathfrak{R}$-трасса, заканчивающаяся в бесконечном множестве состояний.

Возможны два варианта:

1.1. Состояние $\mathtt{s}$ не тау-ограниченное: из состояния $\mathtt{s}$ по τ-маршрутам достижимо бесконечное множество состояний $\mathtt{s`}$. Тогда рассмотрим безопасную $\mathfrak{R}$-трассу $\mu$, заканчивающуюся в состоянии $\mathtt{s}$. Очевидно, она также заканчивается во всех состояниях $\mathtt{s`}$, то есть в бесконечном множестве состояний.

1.2. Состояние $\mathtt{s}$ не БЛК-ветвящееся: в состоянии $\mathtt{s}$ определено бесконечное множество переходов $\mathtt{s}\text{—}\mathtt{z}\rightarrow\mathtt{s`}$ по некоторому внешнему действию $\mathtt{z}$, безопасному после некоторой безопасной $\mathfrak{R}$-трассы μ,

заканчивающейся в состоянии s. Тогда $\mathfrak{R}$-трасса $\mu\cdot\langle z\rangle$ также безопасна и заканчивается в бесконечном множестве постсостояниях s' z-переходов.

2. Достаточность. Если пустая $\mathfrak{R}$-трасса $\mu=\epsilon$ безопасна, то для неё утверждение непосредственно следует из тау-ограниченности начального состояния. Далее по индукции: пусть для некоторой безопасной $\mathfrak{R}$-трассы μ множество $\mathbf{S}$ ***after*** μ конечно. Рассмотрим её безопасные продолжения.

2.1. Продолжение внешним действием z.

Имеем:

$$\mathbf{S}\ \textit{after}\ \mu\cdot\langle z\rangle = \{t\in V_{\mathbf{S}} \mid \exists s\in(\mathbf{S}\ \textit{after}\ \mu)\ \exists s'\ s\text{—}z\rightarrow s'\Rightarrow t\}.$$

Для действия z, безопасного после μ, в каждом из конечного числа состояний $s\in(\mathbf{S}\ \textit{after}\ \mu)$ определено конечное число переходов $s\text{—}z\rightarrow s'$ в силу БЛК-ветвимости состояния s. Значит, число постсостояний s' конечно. Каждое состояние s' достижимо по безопасной $\mathfrak{R}$-трассе $\mu\cdot\langle z\rangle$. Следовательно, множество состояний, достижимых из s' по τ-маршрутам, конечно в силу тау-ограниченности этих состояний. Множество $\mathbf{S}$ ***after*** $\mu\cdot\langle z\rangle$ конечно как объединение конечного множества конечных множеств.

2.2. Продолжение $\mathfrak{R}$-отказом r.

Поскольку отказы – это виртуальные петли в состояниях, имеет место вложение $\mathbf{S}$ ***after*** $\mu\cdot\langle r\rangle \subseteq \mathbf{S}$ ***after*** μ. Поскольку множество $\mathbf{S}$ ***after*** μ конечно, его подмножество тоже конечно.

30. **Доказательство Леммы 14**

Если ***F***-модель содержит трассу $\langle\gamma\rangle$, в ней нет безопасных (***SafeBy*** или ***SafeIn***) трасс. Если трассы $\langle\gamma\rangle$ нет, то безопасная трасса (***SafeBy*** или ***SafeIn***) – это трасса, не заканчивающаяся на дивергенцию и разрушение, в которой каждый встречающийся символ u (внешнее действие или $\mathfrak{R}$-отказ) безопасен (по ***safe by*** или по ***safe in***) после непосредственно предшествующего ему префикса трассы. Безопасность отказа P определяется тем, что кнопка "P" безопасна после трассы. Безопасность действия определяется тем, что оно разрешается некоторой кнопкой, безопасной после трассы.

Условие рефлексивности означает, что безопасность кнопки по ***safe by*** влечёт её безопасность по ***safe in***. Отсюда следует вложенность $\mathbf{\Sigma}\ \textit{SafeBy}\ \mathbf{\Sigma} \subseteq \mathbf{\Sigma}\ \textit{SafeIn}\ \mathbf{\Sigma}$.

Условие транзитивности означает, что безопасность кнопки по ***safe in*** влечёт её безопасность по ***safe by***. Отсюда следует обратная вложенность $\mathbf{\Sigma}\ \textit{SafeBy}\ \mathbf{\Sigma} \supseteq \mathbf{\Sigma}\ \textit{SafeIn}\ \mathbf{\Sigma}$.

31. **Доказательство Леммы 15**

1. Рефлексивность.

По определению отношения ***safe for***,

$$\mathbf{\Sigma}\ \textit{safe for}\ \mathbf{\Sigma} = (\langle\gamma\rangle \notin \mathbf{\Sigma} \Rightarrow \langle\gamma\rangle \notin \mathbf{\Sigma})$$
$$\& \ \forall \sigma \in \mathbf{\Sigma}\ \textit{SafeBy}\ \mathbf{\Sigma} \cap \mathbf{\Sigma}\ \textit{SafeIn}\ \mathbf{\Sigma} \quad \forall P \in \mathfrak{R} \cup \mathfrak{Q}$$
$$(P\ \textit{safe by}\ \mathbf{\Sigma}\ \textit{after}\ \sigma \Rightarrow P\ \textit{safe in}\ \mathbf{\Sigma}\ \textit{after}\ \sigma).$$

Из условия рефлексивности следует истинность импликации.

А тогда получается $\mathbf{\Sigma}\ \textit{safe for}\ \mathbf{\Sigma}$.

По определению отношения ***saco***,

$$\mathbf{\Sigma}\ \textit{saco}\ \mathbf{\Sigma} =_{\text{def}} \mathbf{\Sigma}\ \textit{safe for}\ \mathbf{\Sigma}\ \&$$

$\forall \sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma)\ \ \forall P\ \mathit{safe\ by}\ \Sigma\ \mathit{after}\ \sigma$

$\mathit{obs}(\sigma, P, \Sigma) \subseteq \mathit{obs}(\sigma, P, \Sigma)$.

Поскольку Σ ***safe for*** Σ, имеем Σ ***saco*** Σ.

2. Транзитивность. Пусть $\mathbf{I}$ ***saco*** Σ и Σ ***saco*** Ω.

2.1. Докажем вспомогательное утверждение:

если $\sigma \in \mathit{SafeBy}(\Omega) \cap \mathit{SafeIn}(\mathbf{I})$, то $\sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma)$.

Допустим, это не так.

Поскольку $\sigma \in \mathit{SafeBy}(\Omega)$, $\langle\gamma\rangle \notin \Omega$.

Поскольку Σ ***safe for*** Ω, $\langle\gamma\rangle \notin \Sigma$.

Поскольку $\mathbf{I}$ ***safe for*** Σ, $\langle\gamma\rangle \notin \mathbf{I}$.

Следовательно, $\epsilon \in \mathit{SafeBy}(\Omega) \cap \mathit{SafeIn}(\mathbf{I}) \cap \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma)$.

Значит существует такая трасса μ и такой символ $u \in L_{\mathfrak{R}}$, что

$\mu \cdot \langle u \rangle \in \mathit{SafeBy}(\Omega) \cap \mathit{SafeIn}(\mathbf{I})$ и $\mu \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma)$, но

$\mu \cdot \langle u \rangle \notin \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma)$.

По определению $\mathit{SafeBy}(\Omega)$, имеем u ***safe by*** Ω ***after*** μ.

Следовательно, существует такой отказ $P \in \mathfrak{R} \cup \mathfrak{Q}$, что

P ***safe by*** Ω ***after*** μ и $u=P \in \mathfrak{R}$ или $u \in P$.

Поскольку Σ ***safe for*** Ω,

P ***safe by*** Ω ***after*** $\mu \Rightarrow P$ ***safe in*** Σ ***after*** μ.

По условию Леммы,

P ***safe in*** Σ ***after*** $\mu \Rightarrow P$ ***safe by*** Σ ***after*** μ.

А тогда $\mathbf{I}$ ***saco*** Σ влечёт $\mathit{obs}(\mu, P, \mathbf{I}) \subseteq \mathit{obs}(\mu, P, \Sigma)$.

Поскольку $u \in \mathit{obs}(\mu, P, \mathbf{I})$, имеем $u \in \mathit{obs}(\mu, P, \Sigma)$.

А тогда $\mu \cdot \langle u \rangle \in \Sigma$.

Далее $\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** μ влечёт $\mathtt{u}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** μ.

Тем самым, $\mu\cdot\langle\mathtt{u}\rangle\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})$.

Далее $\mathtt{P}$ ***safe in*** $\boldsymbol{\Sigma}$ ***after*** μ влечёт $\mathtt{u}$ ***safe in*** $\boldsymbol{\Sigma}$ ***after*** μ.

Тем самым, $\mu\cdot\langle\mathtt{u}\rangle\in\boldsymbol{SafeIn}(\boldsymbol{\Sigma})$.

В результате $\mu\cdot\langle\mathtt{u}\rangle\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\boldsymbol{\Sigma})$,

что противоречит допущению.

Тем самым вспомогательное утверждение доказано.

2.2. Теперь докажем, что $\mathbf{I}$ ***safe for*** $\boldsymbol{\Omega}$.

По определению отношения ***safe for***,

$$\begin{aligned}\boldsymbol{\Sigma}\ \textit{safe for}\ \boldsymbol{\Omega} = \quad & (\langle\gamma\rangle\notin\boldsymbol{\Omega}\ \Rightarrow\ \langle\gamma\rangle\notin\boldsymbol{\Sigma})\\ & \&\ \forall\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Omega})\cap\boldsymbol{SafeIn}(\boldsymbol{\Sigma})\ \ \forall\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}\\ & (\mathtt{P}\ \textit{safe by}\ \boldsymbol{\Omega}\ \textit{after}\ \sigma\ \Rightarrow\ \mathtt{P}\ \textit{safe in}\ \boldsymbol{\Sigma}\ \textit{after}\ \sigma),\end{aligned}$$

$$\begin{aligned}\mathbf{I}\ \textit{safe for}\ \boldsymbol{\Sigma} = \quad & (\langle\gamma\rangle\notin\boldsymbol{\Sigma}\ \Rightarrow\ \langle\gamma\rangle\notin\mathbf{I})\\ & \&\ \forall\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I})\ \ \forall\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}\\ & (\mathtt{P}\ \textit{safe by}\ \boldsymbol{\Sigma}\ \textit{after}\ \sigma\ \Rightarrow\ \mathtt{P}\ \textit{safe in}\ \mathbf{I}\ \textit{after}\ \sigma).\end{aligned}$$

Нам нужно доказать, что

$$\begin{aligned}\mathbf{I}\ \textit{safe for}\ \boldsymbol{\Omega} = \quad & (\langle\gamma\rangle\notin\boldsymbol{\Omega}\ \Rightarrow\ \langle\gamma\rangle\notin\mathbf{I})\\ & \&\ \forall\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Omega})\cap\boldsymbol{SafeIn}(\mathbf{I})\ \ \forall\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}\\ & (\mathtt{P}\ \textit{safe by}\ \boldsymbol{\Omega}\ \textit{after}\ \sigma\ \Rightarrow\ \mathtt{P}\ \textit{safe in}\ \mathbf{I}\ \textit{after}\ \sigma).\end{aligned}$$

Имеем:

$\langle\gamma\rangle\notin\boldsymbol{\Omega}\ \Rightarrow\ \langle\gamma\rangle\notin\boldsymbol{\Sigma}$ и $\langle\gamma\rangle\notin\boldsymbol{\Sigma}\ \Rightarrow\ \langle\gamma\rangle\notin\mathbf{I}$, то есть $\langle\gamma\rangle\notin\boldsymbol{\Omega}\ \Rightarrow\ \langle\gamma\rangle\notin\mathbf{I}$.

По вспомогательному утверждению, $\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Omega})\cap\boldsymbol{SafeIn}(\mathbf{I})$

влечёт $\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\boldsymbol{\Sigma})$.

А тогда $\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Omega})\cap\boldsymbol{SafeIn}(\boldsymbol{\Sigma})$ и поэтому $\boldsymbol{\Sigma}$ ***safe for*** $\boldsymbol{\Omega}$ влечёт

$\mathtt{P}$ ***safe by*** $\boldsymbol{\Omega}$ ***after*** $\sigma \Rightarrow \mathtt{P}$ ***safe in*** $\boldsymbol{\Sigma}$ ***after*** σ.

По условию Леммы,

$\mathtt{P}$ ***safe in*** $\boldsymbol{\Sigma}$ ***after*** $\sigma \Rightarrow \mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** σ.

Поскольку $\sigma \in \boldsymbol{SafeBy}(\boldsymbol{\Sigma}) \cap \boldsymbol{SafeIn}(\mathbf{I})$ и $\mathbf{I}$ ***safe for*** $\boldsymbol{\Sigma}$, имеем

$\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** $\sigma \Rightarrow \mathtt{P}$ ***safe in*** $\mathbf{I}$ ***after*** σ.

В результате $\sigma \in \boldsymbol{SafeBy}(\boldsymbol{\Omega}) \cap \boldsymbol{SafeIn}(\mathbf{I})$ влечёт

$\mathtt{P}$ ***safe by*** $\boldsymbol{\Omega}$ ***after*** $\sigma \Rightarrow \mathtt{P}$ ***safe in*** $\mathbf{I}$ ***after*** σ.

Следовательно, $\mathbf{I}$ ***safe for*** $\boldsymbol{\Omega}$.

2.3. Теперь пусть $\sigma \in \boldsymbol{SafeBy}(\boldsymbol{\Omega}) \cap \boldsymbol{SafeIn}(\mathbf{I})$ и $\mathtt{P}$ ***safe by*** $\boldsymbol{\Omega}$ ***after*** σ. Нам осталось показать, что $\boldsymbol{obs}(\sigma, \mathtt{P}, \mathbf{I}) \subseteq \boldsymbol{obs}(\sigma, \mathtt{P}, \boldsymbol{\Omega})$.

Поскольку $\boldsymbol{\Sigma}$ ***safe for*** $\boldsymbol{\Omega}$,

$\mathtt{P}$ ***safe by*** $\boldsymbol{\Omega}$ ***after*** $\sigma \Rightarrow \mathtt{P}$ ***safe in*** $\boldsymbol{\Sigma}$ ***after*** σ.

По вспомогательному утверждению, $\sigma \in \boldsymbol{SafeBy}(\boldsymbol{\Sigma}) \cap \boldsymbol{SafeIn}(\boldsymbol{\Sigma})$.

По условию Леммы,

$\mathtt{P}$ ***safe in*** $\boldsymbol{\Sigma}$ ***after*** $\sigma \Rightarrow \mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** σ.

А тогда $\mathbf{I}$ ***saco*** $\boldsymbol{\Sigma}$ влечёт $\boldsymbol{obs}(\sigma, \mathtt{P}, \mathbf{I}) \subseteq \boldsymbol{obs}(\sigma, \mathtt{P}, \boldsymbol{\Sigma})$.

Кроме того, $\boldsymbol{\Sigma}$ ***saco*** $\boldsymbol{\Omega}$ влечёт $\boldsymbol{obs}(\sigma, \mathtt{P}, \boldsymbol{\Sigma}) \subseteq \boldsymbol{obs}(\sigma, \mathtt{P}, \boldsymbol{\Omega})$.

Тем самым, $\boldsymbol{obs}(\sigma, \mathtt{P}, \mathbf{I}) \subseteq \boldsymbol{obs}(\sigma, \mathtt{P}, \boldsymbol{\Omega})$.

Утверждение доказано.

32. Доказательство Теоремы 17

Безопасность $\mathfrak{R}$-кнопок одинаково определяется по отношениям ***safe by*** и ***safe in***. Поэтому при отсутствии $\mathfrak{Q}$-кнопок эти отношения совпадают. Тем самым, выполнены условия рефлексивности и транзитивности. А тогда, по Лемме 15, отношение ***saco*** рефлексивно и транзитивно.

33. **Доказательство Теоремы 18**

Также, поскольку $\mathfrak{Q}$-кнопок нет, отношения ***safe by*** и ***safe in*** совпадают, и мы будем просто обозначать их как ***safe***.

Следовательно, выполнены условия рефлексивности и транзитивности.

А тогда, по Лемме 14, множества безопасных трасс по этим отношениям совпадают ***SafeBy***=***SafeIn***, и мы будем просто обозначать их как ***Safe***.

1. Покажем, что $\Sigma_1 \sim \Sigma_2 \Rightarrow \mathit{Safe}(\Sigma_1) = \mathit{Safe}(\Sigma_2)$.

В силу симметрии, достаточно показать, что $\mathit{Safe}(\Sigma_1) \subseteq \mathit{Safe}(\Sigma_2)$.

По определению отношения ***safe for***, $\langle\gamma\rangle \in \Sigma_1 \Leftrightarrow \langle\gamma\rangle \in \Sigma_2$.

Наличие γ-трассы ведёт к отсутствию безопасных $\mathfrak{R}$-трасс в обеих ***F***-моделях.

Пусть в обеих ***F***-моделях нет γ-трассы.

Тогда $\epsilon \in \mathit{Safe}(\Sigma_1) \cap \mathit{Safe}(\Sigma_2)$.

Далее по индукции достаточно показать, что, если $\sigma\cdot\langle u\rangle \in \mathit{Safe}(\Sigma_1)$, где $\sigma \in \mathit{Safe}(\Sigma_1) \cap \mathit{Safe}(\Sigma_2)$ и $u \in L_{\mathfrak{R}}$, то $\sigma\cdot\langle u\rangle \in \mathit{Safe}(\Sigma_2)$.

Очевидно, u ***safe*** Σ_1 ***after*** σ.

Тогда Σ_2 ***safe for*** Σ_1 влечёт u ***safe*** Σ_2 ***after*** σ.

Но тогда Σ_1 ***saco*** Σ_2 влечёт $\sigma\cdot\langle u\rangle \in \Sigma_2$ и, тем самым, $\sigma\cdot\langle u\rangle \in \mathit{Safe}(\Sigma_2)$, что и требовалось показать.

2. Покажем, что $\Sigma_1 \sim \Sigma_2 \Rightarrow \mathfrak{I}(\Sigma_1) = \mathfrak{I}(\Sigma_2)$ & $\mathit{Safe}\mathfrak{I}(\Sigma_1) = \mathit{Safe}\mathfrak{I}(\Sigma_2)$.

Первая часть утверждение (равенство классов конформных реализаций) прямо следует из эквивалентности спецификаций по конформности и транзитивности конформности.

Докажем вторую часть утверждения: равенство классов безопасных реализаций. Действительно, если бы это было не так, что нашлась бы такая

реализация $\mathbf{I}$, безопасная, для определённости, для первой спецификации $\mathbf{I}$ ***safe for*** $\boldsymbol{\Sigma_1}$, которая опасна для второй спецификации $\mathbf{I}$ ~~***safe for***~~ $\boldsymbol{\Sigma_2}$. Но тогда либо a) $\langle\gamma\rangle\in\mathbf{I}$, либо b) найдётся такая безопасная $\mathfrak{R}$-трасса $\sigma\in\boldsymbol{Safe}(\boldsymbol{\Sigma_2})$, после которой некоторая $\mathfrak{R}$-кнопка $\mathtt{P}$ ***safe*** $\boldsymbol{\Sigma_2}$ ***after*** σ, но $\mathtt{P}$ ~~***safe***~~ $\mathbf{I}$ ***after*** σ. В случае a) $\mathbf{I}$ ~~***safe for***~~ $\boldsymbol{\Sigma_1}$, что не верно. В случае b) найдётся такое действие $\mathtt{z}\in\mathtt{P}$, что $\sigma\cdot\langle\mathtt{z},\gamma\rangle\in\mathbf{I}$. Поскольку, по доказанному, $\boldsymbol{Safe}(\boldsymbol{\Sigma_1})=\boldsymbol{Safe}(\boldsymbol{\Sigma_2})$, имеем $\sigma\in\boldsymbol{Safe}(\boldsymbol{\Sigma_1})$. Но тогда, поскольку $\mathbf{I}$ ***safe for*** $\boldsymbol{\Sigma_1}$, не может быть $\mathtt{P}$ ***safe*** $\boldsymbol{\Sigma_1}$ ***after*** σ, что противоречит $\boldsymbol{\Sigma_1}$ ***safe for*** $\boldsymbol{\Sigma_2}$.

34. Доказательство Леммы 16

1. Сначала докажем, что $\mathbf{I}$ ***safe for*** $\boldsymbol{\Sigma}$. Допустим, утверждение не верно.

Если $\langle\gamma\rangle\in\mathbf{I}$, то $\langle\gamma\rangle\in\boldsymbol{\Sigma}$ и $\mathbf{I}$ ***saco*** $\boldsymbol{\Sigma}$.

Пусть $\langle\gamma\rangle\notin\mathbf{I}$.

Тогда существует такая трасса $\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I})$ и такая кнопка “$\mathtt{P}$”, где $\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}$, что $\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** σ, но $\mathtt{P}$ ~~***safe in***~~ $\mathbf{I}$ ***after*** σ.

Тогда $\mathtt{P}\notin\mathfrak{R}$ & $\sigma\cdot\langle\mathtt{P}\rangle\in\mathbf{I}\ \vee\ \exists\mathtt{z}\in\mathtt{P}\ \sigma\cdot\langle\mathtt{z},\gamma\rangle\in\mathbf{I}\ \vee\ \sigma\cdot\langle\Delta\rangle\in\mathbf{I}$.

Поскольку нет $\mathfrak{Q}$-кнопок, остаётся $\exists\mathtt{z}\in\mathtt{P}\ \sigma\cdot\langle\mathtt{z},\gamma\rangle\in\mathbf{I}\ \vee\ \sigma\cdot\langle\Delta\rangle\in\mathbf{I}$.

Поскольку $\mathbf{I}\subseteq\boldsymbol{\Sigma}$, $\exists\mathtt{z}\in\mathtt{P}\ \sigma\cdot\langle\mathtt{z},\gamma\rangle\in\boldsymbol{\Sigma}\ \vee\ \sigma\cdot\langle\Delta\rangle\in\boldsymbol{\Sigma}$.

А это противоречит $\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** σ.

2. Теперь докажем, что $\mathbf{I}$ ***saco*** $\boldsymbol{\Sigma}$. Допустим, утверждение не верно.

Тогда существует такая трасса $\sigma\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I})$ и такая кнопка “$\mathtt{P}$”, где $\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}$, что $\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** σ, но $\boldsymbol{obs}(\sigma,\mathtt{P},\mathbf{I})\not\subseteq\boldsymbol{obs}(\sigma,\mathtt{P},\boldsymbol{\Sigma})$.

Тогда $\exists u\ (u\in P \vee u=P\ \&\ P\in\mathfrak{R})\ \&\ \sigma\cdot\langle u\rangle\in\mathbf{I}\setminus\mathbf{\Sigma}$.

А это противоречит $\mathbf{I}\subseteq\mathbf{\Sigma}$.

35. **Доказательство Леммы 17**

Пусть $\mathbf{I}\in\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$.

Поскольку $\mathbf{\Sigma}`=\cup_{\circ}\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$, $\mathbf{I}\subseteq\mathbf{\Sigma}`$.

Тогда, по Лемме 16, $\mathbf{I}$ ***saco***$_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}$ $\mathbf{\Sigma}`$.

Следовательно, $\mathbf{I}\in\mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\mathbf{\Sigma}`)$, что и следовало доказать.

36. **Доказательство Леммы 18**

1. Сначала докажем, что $\mathbf{\Sigma}`$ ***safe***$_{\mathfrak{R}/\mathfrak{Q}}$ ***for*** $\mathbf{\Sigma}$. Пусть утверждение не верно.

Если $\langle\gamma\rangle\in\mathbf{\Sigma}`$, то найдётся такая конформная реализация $\mathbf{I}$ ***saco***$_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$, что $\langle\gamma\rangle\in\mathbf{I}$. А тогда $\langle\gamma\rangle\in\mathbf{\Sigma}$ и $\mathbf{\Sigma}`$ ***safe***$_{\mathfrak{R}/\mathfrak{Q}}$ ***for*** $\mathbf{\Sigma}$.

Пусть $\langle\gamma\rangle\notin\mathbf{I}$.

Тогда существует такая трасса $\sigma\in$ ***SafeBy***$(\mathbf{\Sigma})\cap$***SafeIn***$(\mathbf{\Sigma}`)$ и такая кнопка “P”, где $P\in\mathfrak{R}\cup\mathfrak{Q}$, что P ***safe by*** $\mathbf{\Sigma}$ ***after*** $\sigma$, но P ~~***safe in***~~ $\mathbf{\Sigma}`$ ***after*** σ.

Тогда $P\notin\mathfrak{R}$ & $\sigma\cdot\langle P\rangle\in\mathbf{\Sigma}`\ \vee\ \exists z\in P\ \sigma\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}`\ \vee\ \sigma\cdot\langle\Delta\rangle\in\mathbf{\Sigma}`$.

Поскольку $\mathbf{\Sigma}`=\cup_{\circ}\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$,

$\exists\mathbf{I}\in\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})\ \ P\notin\mathfrak{R}\ \&\ \sigma\cdot\langle P\rangle\in\mathbf{I}\ \vee\ \exists z\in P\ \sigma\cdot\langle z,\gamma\rangle\in\mathbf{I}\ \vee\ \sigma\cdot\langle\Delta\rangle\in\mathbf{I}$.

А тогда $\mathbf{I}$ ~~***safe for***~~ $\mathbf{\Sigma}$, что не верно.

2. Теперь докажем, что $\mathbf{\Sigma}`$ ***saco***$_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$. Допустим, утверждение не верно.

Тогда существует такая трасса $\sigma \in \mathit{SafeBy}(\Sigma) \cap \mathit{SafeIn}(\Sigma`)$ и такая кнопка "P", где $P \in \mathfrak{R} \cup \mathfrak{Q}$, что $P$ ***safe by*** $\Sigma$ ***after*** $\sigma$, но $\mathit{obs}(\sigma, P, \Sigma`) \not\subseteq \mathit{obs}(\sigma, P, \Sigma)$.

Тогда $\exists u\ (u \in P \vee u = P \ \&\ P \in \mathfrak{R})\ \&\ \sigma \cdot \langle u \rangle \in \Sigma` \setminus \Sigma$.

Поскольку $\Sigma` = \cup_{\circ} \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)$, $\exists I \in \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma)\ \ \sigma \cdot \langle u \rangle \in I \setminus \Sigma$.

А тогда I ~~***saco***~~ Σ, что не верно.

37. Доказательство Леммы 19

P ~~***safe in***~~ I ***after*** μ

$\Rightarrow$ /* по определению ***safe in*** */

$(P \in \mathfrak{Q}\ \&\ \mu \cdot \langle P \rangle \in I) \vee \exists z \in P\ \mu \cdot \langle z, \gamma \rangle \in I \vee \mu \cdot \langle \Delta \rangle \in I$

$\Rightarrow$ /* по замкнутости и полноте модели для $\mu` \in \mathit{di}(\mu)$ */

$(P \in \mathfrak{Q}\ \&\ \mu` \cdot \langle P \rangle \in I) \vee \exists z \in P\ \mu` \cdot \langle z, \gamma \rangle \in I \vee \mu` \cdot \langle \Delta \rangle \in I$

$\Rightarrow$ /* по определению ***safe in*** */

P ~~***safe in***~~ I ***after*** $\mu`$.

38. Доказательство Леммы 20

Сначала покажем, что $T \in \mathit{FMODEL}(L)$, где $T = \Sigma(M, \gamma)$ или $T = \Sigma(M, \Delta)$.

Поскольку вместе с добавляемой трассой $\mu` \cdot \langle z, \gamma \rangle$ или $\mu` \cdot \langle z, \Delta \rangle$ мы добавляем и её, быть может, отсутствовавший ранее префикс $\mu` \cdot \langle z \rangle$ (остальные префиксы были в Σ), множество T является деревом.

1. Допустимость. Достаточно показать допустимость добавленных трасс.

Любая добавленная трасса является допустимой трассой $\mu` \cdot \langle z \rangle$,

заканчивающей действием, или её продолжением разрушением или дивергенцией $\mu`\cdot\langle z,\gamma\rangle$ или $\mu`\cdot\langle z,\Delta\rangle$, которое также допустимо.

2. Согласованность. Достаточно показать согласованность добавленных трасс. Любая добавленная трасса является согласованной трассой $\mu`\cdot\langle z\rangle$, заканчивающей действием, или её продолжением разрушением или дивергенцией $\mu`\cdot\langle z,\gamma\rangle$ или $\mu`\cdot\langle z,\Delta\rangle$, которое также согласовано.

3. Конвергентность сохраняется при добавлении трасс, поскольку мы добавляем только продолжения уже имеющих трасс действием и, возможно, далее разрушением или дивергенцией.

4. Замкнутость. Объединение замкнутых по ***d***-операции множеств замкнуто. Множество $\mathbf{T}$ является объединением множеств $\mathbf{\Sigma}$ и $\boldsymbol{c}(\mu,z,\gamma)$ или $\boldsymbol{c}(\mu,z,\Delta)$, где пара (μ,z) пробегает множество $\mathbf{M}$. Множество $\mathbf{\Sigma}$ замкнуто как ***F***-модель. Осталось показать замкнутость каждого из множеств $\boldsymbol{c}(\mu,z,\gamma)$ или $\boldsymbol{c}(\mu,z,\Delta)$. Действительно, если трасса $\mu`\cdot\langle z\rangle$ согласована, то для любой трассы $\mu``\in\boldsymbol{d}(\mu`)$ трасса $\mu``\cdot\langle z\rangle$ также согласована. Поэтому вместе с добавляемой трассой $\mu`\cdot\langle z\rangle$ и $\mu`\cdot\langle z,\gamma\rangle$ или $\mu`\cdot\langle z,\Delta\rangle$ добавляется трасса $\mu``\cdot\langle z\rangle$ и $\mu``\cdot\langle z,\gamma\rangle$ или $\mu``\cdot\langle z,\Delta\rangle$.

5. Полнота. Нам достаточно показать замкнутость по ***i***-операции для добавленных трасс $\mu`\cdot\langle z\rangle$, $\mu`\cdot\langle z,\gamma\rangle$ или $\mu`\cdot\langle z,\Delta\rangle$. Пусть $\mu`=\mu_1\cdot\mu_2$, где μ_1 в $\mathbf{T}$ заканчивается отказом и не продолжается действиями из отказа P. Отсюда следует, что для любой добавленной трассы $\mu_1\cdot\langle z_1\rangle$ должно быть $z_1\notin P$. В частности, для $\mu_2=\epsilon$ и $z_1=z$. А тогда согласованность трассы $\mu`\cdot\langle z\rangle=\mu_1\cdot\mu_2\cdot\langle z\rangle$ влечёт согласованность трассы $\mu_1\cdot\langle P\rangle\cdot\mu_2\cdot\langle z\rangle$ и,

следовательно, добавление трасс $\mu_1\cdot\langle P\rangle\cdot\mu_2\cdot\langle z\rangle$, $\mu_1\cdot\langle P\rangle\cdot\mu_2\cdot\langle z,\gamma\rangle$ или $\mu_1\cdot\langle P\rangle\cdot\mu_2\cdot\langle z,\Delta\rangle$.

Нам осталось показать, что для каждой пары $(\mu,z)\in\mathbf{M}$ имеет место $\mu\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}(\mathbf{M},\gamma)$ и $\mu\cdot\langle z,\Delta\rangle\in\mathbf{\Sigma}(\mathbf{M},\Delta)$. Это непосредственно следует из согласованности трассы $\mu\cdot\langle z\rangle$ и $\mu\in\boldsymbol{d}(\mu)$.

39. **Доказательство Леммы 21**

Пусть $\mu\in\mathbf{\Sigma}`^{\downarrow L_{\mathfrak{R}}}$, $z\in L$, $z\notin\cup\circ\boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ и $z$ ***~~safe in~~*** $\mathbf{\Sigma}`$ ***after*** μ.

Рассмотрим множество трасс $\mathbf{\Sigma}``=\mathbf{\Sigma}`(\{(\mu,z)\},\gamma)$.

По Лемме 20, оно является ***F***-моделью.

Покажем, что $\mathbf{\Sigma}``=\mathbf{\Sigma}`$.

Поскольку, очевидно, $\mathbf{\Sigma}`\subseteq\mathbf{\Sigma}``$, нам достаточно показать, что $\mathbf{\Sigma}``\subseteq\mathbf{\Sigma}`$.

Поскольку $\mathbf{\Sigma}`=\cup\circ\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$, нам достаточно показать, что $\mathbf{\Sigma}``$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$.

Допустим, это не так.

По Лемме 18, $\mathbf{\Sigma}`$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$.

Поэтому после добавления трасс вида $\mu`\cdot\langle z\rangle$ и $\mu`\cdot\langle z,\gamma\rangle$, где трасса $\mu`$ была раньше, нарушение конформности может означать только нарушение безопасности на одной из трасс $\mu`\cdot\langle z\rangle$.

Это значит, что для некоторой кнопки “P” имеет место: $z\in P$ & P $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\mathbf{\Sigma}$ ***after*** $\mu`$.

Поскольку $\mathbf{\Sigma}`$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$, имеем $P$ ***safe in*** $\mathbf{\Sigma}`$ ***after*** $\mu`$.

А тогда, поскольку $\mu`\in\boldsymbol{di}(\mu)$, по Лемме 19, $P$ ***safe in*** $\mathbf{\Sigma}`$ ***after*** μ.

Но это для $z\in P$ противоречит z ***~~safe in~~*** $\mathbf{\Sigma}`$ ***after*** μ.

40. **Доказательство Леммы 22**

Во всех трёх правилах для безопасности кнопки по ***safe by*** после $\mathfrak{R}$-трассы необходимо, чтобы трасса не продолжалась дивергенцией. Но это же требуется для безопасности кнопки по ***safe in***.

1. Первое правило протокола определяет безопасные $\mathfrak{R}$-кнопки как неразрушающие, что совпадает с безопасность этих кнопок по отношению ***safe in***.

2. Второе правило требует, чтобы внешнее действие z, продолжающее трассу μ и разрешаемое неразрушающей кнопкой "T", разрешалось какой-нибудь безопасной кнопкой "P". Если это не так, то z ~~***safe in***~~ $\boldsymbol{\Sigma`}$ ***after*** $\mu$. А тогда, по Лемме 21 $\mu\cdot\langle z,\gamma\rangle\in\boldsymbol{\Sigma`}$, что противоречит наличию неразрушающей кнопки "T", разрешающей действие z.

3. Третье правило требует, чтобы безопасная $\mathfrak{Q}$-кнопка "Q" была неразрушающей и разрешала некоторое внешнее действие $z\in Q$, которым трасса продолжается. По отношению ***safe in*** безопасная $\mathfrak{Q}$-кнопка "Q" также неразрушающая и не продолжается отказом Q, что влечёт, по конвергентности ***F***-модели, продолжение некоторым действием $z\in Q$.

41. **Доказательство Леммы 23**

Если $\langle\gamma\rangle\in\boldsymbol{\Sigma`}$, то такой спецификации конформна любая реализация как в $\mathfrak{R}\cup\mathfrak{Q}/\varnothing$-, так и в $\mathfrak{R}/\mathfrak{Q}$-семантике. Поэтому $\boldsymbol{\mathfrak{I}}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\boldsymbol{\Sigma`})=\boldsymbol{\mathfrak{I}}_{\mathfrak{R}/\mathfrak{Q}}(\boldsymbol{\Sigma`})$.

Далее будем считать, что $\langle\gamma\rangle\notin\boldsymbol{\Sigma`}$.

Пусть $\mathbf{I}\in\boldsymbol{\mathfrak{I}}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\boldsymbol{\Sigma`})$, то есть $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}$ $\boldsymbol{\Sigma`}$.

Нам надо показать, что $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\boldsymbol{\Sigma`}$.

1. Сначала покажем: $\forall\sigma\in \mathit{SafeBy}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma`)\cap \mathit{SafeBy}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\Sigma`)\quad \forall P\in\mathfrak{R}\cup\mathfrak{Q}$

$(P\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma \Rightarrow P\ \mathit{safe}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma)$.

Действительно,

$P\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma$

$\Rightarrow \forall u\in P\ \sigma\cdot\langle u,\gamma\rangle\notin\Sigma` \ \&\ \sigma\cdot\langle\Delta\rangle\notin\Sigma`$

$\Rightarrow P\ \mathit{safe}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma$.

2. Из **1.** следует, что $\mathit{SafeBy}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma`)\subseteq \mathit{SafeBy}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\Sigma`)$.

3. Докажем, что $\mathbf{I}\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{for}\ \Sigma`$. Пусть утверждение не верно.

По Лемме 6, существует такая трасса $\sigma\in \mathit{SafeBy}_{\mathfrak{R}/\mathfrak{Q}}(\Sigma`)\cap\mathbf{I}$ и такая кнопка "P", где $P\in\mathfrak{R}\cup\mathfrak{Q}$, что $P\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma$, но $P\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{in}\ \mathbf{I}\ \mathit{after}\ \sigma$.

Тогда $P\notin\mathfrak{R}\ \&\ \sigma\cdot\langle P\rangle\in\mathbf{I} \vee \exists z\in P\ \sigma\cdot\langle z,\gamma\rangle\in\mathbf{I} \vee \sigma\cdot\langle\Delta\rangle\in\mathbf{I}$.

По 1. и 2., $\sigma\in \mathit{SafeBy}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\Sigma`)$ и $P\ \mathit{safe}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma$.

Поскольку $\mathbf{I}\ \mathit{safe}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathit{for}\ \Sigma`$, должно быть $P\ \mathit{safe}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \mathit{in}\ \mathbf{I}\ \mathit{after}\ \sigma$.

А тогда $\forall z\in P\ \sigma\cdot\langle z,\gamma\rangle\notin\mathbf{I}\ \&\ \sigma\cdot\langle\Delta\rangle\notin\mathbf{I}$.

Поэтому остаётся $P\notin\mathfrak{R}\ \&\ \sigma\cdot\langle P\rangle\in\mathbf{I}$.

Поскольку $\mathbf{I}\ \mathit{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}\ \Sigma`$, должно быть $\sigma\cdot\langle P\rangle\in\Sigma`$.

Поскольку $P\notin\mathfrak{R}$, должно быть $P\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{in}\ \Sigma`\ \mathit{after}\ \sigma$.

По условию рефлексивности, $\mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{by}\ \Sigma` \Rightarrow \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{in}\ \Sigma`$.

Следовательно, $P\ \mathit{safe}_{\mathfrak{R}/\mathfrak{Q}}\ \mathit{by}\ \Sigma`\ \mathit{after}\ \sigma$.

Мы пришли к противоречию, и утверждение **3.** доказано.

4. Теперь докажем, что $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\boldsymbol{\Sigma`}$. Допустим, утверждение не верно.

По Лемме 8, существует такая трасса $\sigma\in\boldsymbol{SafeBy}_{\mathfrak{R}/\mathfrak{Q}}(\boldsymbol{\Sigma`})\cap\mathbf{I}$ и такая кнопка "P", что $\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}$, $\mathtt{P}$ $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\boldsymbol{\Sigma`}$ ***after*** σ, но $\boldsymbol{obs}(\sigma,\mathtt{P},\mathbf{I})\not\subseteq\boldsymbol{obs}(\sigma,\mathtt{P},\boldsymbol{\Sigma`})$.

По 1. и 2., $\sigma\in\boldsymbol{SafeBy}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\boldsymbol{\Sigma`})$ и $\mathtt{P}$ $\boldsymbol{safe}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}$ ***by*** $\boldsymbol{\Sigma`}$ ***after*** σ.

А тогда, поскольку $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}$ $\boldsymbol{\Sigma`}$, имеем $\boldsymbol{obs}(\sigma,\mathtt{P},\mathbf{I})\subseteq\boldsymbol{obs}(\sigma,\mathtt{P},\boldsymbol{\Sigma`})$, что противоречит допущению.

42. Доказательство Леммы 24

Пусть для $\boldsymbol{\Sigma`}$ выполнено условие транзитивности.

Пусть $\mathbf{I}\in\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\boldsymbol{\Sigma`})$, то есть $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\boldsymbol{\Sigma`}$.

По утверждению Леммы 18, $\boldsymbol{\Sigma`}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\boldsymbol{\Sigma}$.

Поскольку выполнено условие транзитивности, по Лемме 15, $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\boldsymbol{\Sigma}$, то есть $\mathbf{I}\in\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\boldsymbol{\Sigma})$, что и требовалось доказать.

43. Доказательство Теоремы 19

Отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\boldsymbol{\Sigma`}$ ***after*** $\sigma$ = ***safe in*** $\boldsymbol{\Sigma`}$ ***after*** σ для $\sigma\in\boldsymbol{SafeIn}(\boldsymbol{\Sigma`})$, по Лемме 22, удовлетворяет всем трём правилам протокола взаимодействия.

Такое отношение $\boldsymbol{safe}_{\mathfrak{R}/\mathfrak{Q}}$ ***by*** $\boldsymbol{\Sigma`}$, очевидно, удовлетворяет условиям рефлексивности и транзитивности. Поэтому:

по Лемме 17, $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\boldsymbol{\Sigma})\subseteq\mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\boldsymbol{\Sigma`})$,

по Лемме 23, $\mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\boldsymbol{\Sigma`})\subseteq\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\boldsymbol{\Sigma`})$,

по Лемме 24, $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma`}) \subseteq \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$.

Тем самым, $\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma}) = \mathfrak{I}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}(\mathbf{\Sigma`}) = \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma`})$.

44. Доказательство Теоремы 20

Утверждение непосредственно следует из Теоремы 17 и Теоремы 19.

45. Доказательство Теоремы 21

$\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma`}$ $\Rightarrow$ /* по Теореме 19 */ $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma}$

$\Rightarrow$ /* по определению $\mathfrak{I}$ */ $\mathbf{I} \in \mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$

$\Rightarrow$ /* $\mathbf{\Sigma`} = \cup_{\circ}\mathfrak{I}_{\mathfrak{R}/\mathfrak{Q}}(\mathbf{\Sigma})$ */ $\mathbf{I} \subseteq \mathbf{\Sigma`}$.

$\mathbf{I} \subseteq \mathbf{\Sigma`}$ $\Rightarrow$ /* по Лемме 16 */ $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}\cup\mathfrak{Q}/\varnothing}$ $\mathbf{\Sigma`}$

$\Rightarrow$ /* по Теореме 19 */ $\mathbf{I}$ $\boldsymbol{saco}_{\mathfrak{R}/\mathfrak{Q}}$ $\mathbf{\Sigma`}$.

46. Доказательство Леммы 25

Преобразование "$\Rightarrow$" выполняется на LTS $\mathbf{s}$ простой заменой алфавита $\mathtt{L}$ на алфавит $\mathtt{L}^{\Rightarrow}$. Очевидно, при этом получается LTS-модель. Для $\boldsymbol{F}$-модели преобразование определяется как $\mathbf{I}^{\Rightarrow} = \boldsymbol{F}(\boldsymbol{F2L}(\mathbf{I})^{\Rightarrow})$. Отсюда, в силу Теоремы 12 и Теоремы 9, непосредственно следует, что $\mathbf{I}^{\Rightarrow} \in \boldsymbol{FMODEL}(\mathtt{L}^{\Rightarrow})$.

Обратное преобразование "$\Leftarrow$" эквивалентно обратной замене в LTS $\boldsymbol{F2L}(\mathbf{I})^{\Rightarrow}$ алфавита $\mathtt{L}^{\Rightarrow}$ на алфавит $\mathtt{L}$, в результате чего получается исходное множество $\boldsymbol{F}$-трасс:

$\mathbf{I}^{\Rightarrow\Leftarrow} = \boldsymbol{F}(\boldsymbol{F2L}(\mathbf{I})^{\Rightarrow})^{\Leftarrow} = \boldsymbol{F}(\boldsymbol{F2L}(\mathbf{I})) = \mathbf{I}$.

47. Доказательство Леммы 26

Множество финальных трасс является деревом, по построению.

1. Допустимость. По правилам вывода, если финальная трасса $\mu^{\rightarrow}$ заканчивается дивергенцией или разрушением, то $\mu^{\rightarrow}\in\boldsymbol{\Sigma_1}$ и, следовательно, она ничем не продолжается.

2. Согласованность. По правилам вывода, дивергенция и разрушение могут следовать только после не-отказа, то есть не могут следовать после отказа. Продолжение не-отказом, действием из $\mathtt{L}$ или $\mathfrak{R}$-отказом регулируется определениями $\boldsymbol{G}(\mu)$, $\boldsymbol{D}(\mu)$ и $\boldsymbol{U}(\mu)$, в которых явно требуется согласованность продолженной финальной трассы.

3. $\mathfrak{R}$-конвергентность. По правилам вывода, финальные трассы из множества $\boldsymbol{\Sigma_1}$ заканчиваются или продолжаются дивергенцией или разрушением, поэтому не обязаны быть конвергентными.

Пусть финальная трасса $\mu^{\rightarrow}\in\boldsymbol{\Sigma_0}$ и рассмотрим $\mathfrak{R}$-отказ $\mathtt{P}$.

Сначала рассмотрим случай $\mathtt{P}\in\boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ и покажем, что $\mu^{\rightarrow}\cdot\langle\mathtt{P}^{\rightarrow}\rangle\in\boldsymbol{\Sigma_0}$.

Нам надо показать, что $\mathtt{P}\in\boldsymbol{U}(\mu)$, то есть согласованность и конформность трассы $\mu\cdot\langle\mathtt{P}\rangle$ и безопасность отказа $\mathtt{P}$ после трассы μ.

Согласованность трассы $\mu^{\rightarrow}$ влечёт согласованность трассы μ. Продолжение отказом не может нарушить согласованность, поэтому из согласованности трассы μ следует согласованность трассы $\mu\cdot\langle\mathtt{P}\rangle$.

Далее представим трассу в виде $\mu=\mu_1\cdot\langle\mathtt{P}\rangle\cdot\rho$, где ρ трасса отказов. Тогда $\mathtt{P}\in\boldsymbol{safe}(\mu_1)$, следовательно, найдётся такая трасса $\lambda\in\boldsymbol{drt}(\mu_1)$, что $\lambda\cdot\langle\mathtt{P}\rangle\in\boldsymbol{tt}^{\#}(\boldsymbol{\Sigma})$. Поскольку ρ трасса отказов, $\lambda\in\boldsymbol{drt}(\mu)$. Тем самым выполнено условие $\mathtt{P}\in\boldsymbol{safe}(\mu)$.

Наконец, покажем, что $\mu\cdot\langle\mathtt{P}\rangle$ ***conf***. Допустим это не так. Тогда должна найтись такая трасса $\lambda\in\boldsymbol{drt}(\mu\cdot\langle\mathtt{P}\rangle)$ и такая трасса κ, что $\kappa\cdot\langle\mathtt{P}\rangle\le\lambda$ и

$\kappa\cdot\langle \mathtt{P}\rangle \in \boldsymbol{tt}^{\#}(\boldsymbol{\Sigma}) \setminus \boldsymbol{\Sigma}$. Однако в случае $\mathtt{P} \in \boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ имеет место равенство $\boldsymbol{drt}(\mu\cdot\langle \mathtt{P}\rangle) = \boldsymbol{drt}(\mu)$, и поэтому наличие такой трассы λ противоречит конформности трассы μ.

Теперь рассмотрим случай $\mathtt{P} \notin \boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$ и покажем, что $\mu^{\rightarrow}\cdot\langle ₽\rangle \in \boldsymbol{\Sigma_1}$.

Действительно, продолжение трассы не-отказом регулируется определениями $\boldsymbol{G}(\mu)$ и $\boldsymbol{D}(\mu)$, которые покрывают все случаи, когда $\mathtt{P} \notin \boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$: если $₽ \notin \boldsymbol{safe}(\mu)$, то трасса продолжается не-отказом ₽ и далее разрушением (правило вывода 3); если $₽ \in \boldsymbol{safe}(\mu)$, то трасса продолжается не-отказом ₽ и далее дивергенцией (правило вывода 4).

4. $\mathfrak{R}$-полнота. Поскольку определения $\boldsymbol{G}(\mu)$ и $\boldsymbol{D}(\mu)$ покрывают все случаи, когда $\mathtt{P} \notin \boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$, для каждого $\mathfrak{R}$-отказа P финальная трасса $\mu^{\rightarrow}$ не продолжается не-отказом ₽ только в том случае, когда $\mathtt{P} \in \boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$.

Нам надо показать, что для любой финальной трассы $\mu^{\rightarrow}\cdot\lambda$, где $\mathtt{P} \in \boldsymbol{Im}\circ\boldsymbol{postf}(\mu)$, трасса $\mu^{\rightarrow}\cdot\langle \mathtt{P}^{\rightarrow}\rangle\cdot\lambda$ также будет финальной. Допустим, это не так. Тогда трассу λ можно представить в виде $\lambda=\lambda_1\cdot\langle \mathtt{u}\rangle\cdot\lambda_2$ так, что трасса $\mu^{\rightarrow}\cdot\langle \mathtt{P}^{\rightarrow}\rangle\cdot\lambda_1$ финальная, а её продолжение символом u не финально. Поскольку перед трассой λ_1 располагается отказ $\mathtt{P}^{\rightarrow}$, эта трасса, по правилам вывода, не может начинаться с разрушения или дивергенции. Учитывая допустимость финальных трасс, возможны следующие варианты:

a. $\lambda_1=\pi^{\rightarrow}$ и u=⦵,

b. $\lambda_1=\pi^{\rightarrow}\cdot\langle ⦵\rangle$ и $\mathtt{u}=\gamma$,

c. $\lambda_1=\pi^{\rightarrow}\cdot\langle ⦵\rangle$ и $\mathtt{u}=\Delta$,

d. $\lambda_1=\pi^{\rightarrow}$ и u=z или

e. $\lambda_1=\pi^{\rightarrow}$ и u=Q,

где $\mathtt{Q}$ отказ, а $\mathtt{z}$ действие из $\mathtt{L}$.

a. Согласованность трассы $\mu\cdot\pi\cdot\langle\Theta\rangle$ влечёт согласованность трассы $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\Theta\rangle$, поскольку $\mathtt{P}\in\boldsymbol{Im}_{\circ}\boldsymbol{postf}(\mu)$. Следовательно, $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\Theta\rangle\in\boldsymbol{\Sigma_1}$, что противоречит допущению.

b. Поскольку $\mathtt{P}\in\boldsymbol{Im}_{\circ}\boldsymbol{postf}(\mu)$, имеет место равенство $\boldsymbol{drt}(\mu\cdot\langle\mathtt{P}\rangle\cdot\pi)=\boldsymbol{drt}(\mu\cdot\pi)$. Следовательно, $\Theta\notin\boldsymbol{safe}(\mu\cdot\pi)$ влечёт $\Theta\notin\boldsymbol{safe}(\mu\cdot\langle\mathtt{P}\rangle\cdot\pi)$ и $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\Theta,\gamma\rangle\in\boldsymbol{\Sigma_1}$, что противоречит допущению.

c. Аналогично варианту b, $\Theta\in\boldsymbol{safe}(\mu\cdot\pi)$ влечёт $\Theta\in\boldsymbol{safe}(\mu\cdot\langle\mathtt{P}\rangle\cdot\pi)$ и $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\Theta,\Delta\rangle\in\boldsymbol{\Sigma_1}$, что противоречит допущению.

d. Аналогично варианту b, трасса $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\mathtt{z}\rangle$ согласована и $\mathtt{z}\in\boldsymbol{safe}(\mu\cdot\langle\mathtt{P}\rangle\cdot\pi)$. Также, поскольку $\boldsymbol{drt}(\mu\cdot\langle\mathtt{P}\rangle\cdot\pi)=\boldsymbol{drt}(\mu\cdot\pi)$, условие $\mu\cdot\pi\cdot\langle\mathtt{z}\rangle$ ***conf*** влечёт $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\mathtt{z}\rangle$ ***conf***. Следовательно, $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\mathtt{z}\rangle\in\boldsymbol{\Sigma_0}$, что противоречит допущению.

e. Аналогично варианту d, трасса $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\mathtt{Q}\rangle$ согласована и $\mathtt{Q}\in\boldsymbol{safe}(\mu\cdot\langle\mathtt{P}\rangle\cdot\pi)$, а условие $\mu\cdot\pi\cdot\langle\mathtt{Q}\rangle$ ***conf*** влечёт $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\mathtt{Q}\rangle$ ***conf***. Следовательно, $\mu\cdot\langle\mathtt{P}\rangle\cdot\pi\cdot\langle\mathtt{Q}\rangle\in\boldsymbol{\Sigma_0}$, что противоречит допущению.

Итак, мы пришли к противоречию и, следовательно, наше допущение не верно, а $\mathfrak{R}$-полнота доказана.

48. Доказательство Леммы 27

d-замыкание дерева трасс является деревом. Свойства допустимости, согласованности, $\mathfrak{R}$-конвергентности, очевидно, сохраняются. Также

сохраняется $\mathfrak{R}$-полнота (доказательство смотри в Доказательстве Теоремы 3, в п.6). Замкнутость после ***d***-замыкания очевидна. Далее используется Лемма 26.

49. Доказательство Леммы 28

По Лемме 27, $\cup\circ\boldsymbol{d}\circ\boldsymbol{final}(\Sigma)$ является $\mathfrak{R}^{\rightarrow}$-моделью в алфавите $\mathtt{L}^{\Rightarrow}$.

$\mathfrak{R}^{\rightarrow}$-трассы не содержат $\mathfrak{Q}^{\rightarrow}$-отказов.

Поскольку $\boldsymbol{Comp}=\boldsymbol{Ext}\circ\cup\circ\boldsymbol{d}\circ\boldsymbol{final}$, то, по Теореме 3, $\boldsymbol{Comp}(\Sigma)$ является ***F***-моделью в алфавите $\mathtt{L}^{\Rightarrow}$.

По определению, $\boldsymbol{Ext}(\Sigma)=\cup\circ\boldsymbol{d}\circ\cup\circ\boldsymbol{i}_{\mathcal{P}(\mathtt{L})}\circ\cup\circ\boldsymbol{e}(\Sigma)$.

По правилам вывода, финальные трассы не содержат $\mathfrak{Q}^{\rightarrow}$-отказов.

Также, по правилам вывода и определению $\boldsymbol{G}(\mu)$ и $\boldsymbol{D}(\mu)$, множество финальных трасс обладает следующим свойством: финальная трасса не продолжается не-отказом $\ominus$ только в том случае, когда она a) продолжается или заканчивается разрушением или дивергенцией, или b) когда она содержит отказ $\mathtt{Q}$ в постфиксе отказов. Для $\mathtt{Q}\in\mathfrak{Q}^{\rightarrow}$ остаётся только случай a).

По определению операции ***e*** (вставка пустых отказов), она сохраняет это свойство множества трасс.

А тогда операция $\boldsymbol{i}_{\mathcal{P}(\mathtt{L})}$ не вставляет $\mathfrak{Q}^{\rightarrow}$-отказы.

Наконец, ***d***-операция только удаляет отказы.

Тем самым, в трассах $\boldsymbol{Comp}(\Sigma)$ нет $\mathfrak{Q}^{\rightarrow}$-отказов.

50. Доказательство Леммы 29

1. По Лемме 28, трассы ***F***-модели $\boldsymbol{Comp}(\Sigma)$ не содержат $\mathfrak{Q}^{\rightarrow}$-отказов. Отсюда непосредственно следует утверждение 1).

2. Из отсутствия $\mathfrak{Q}^{\rightarrow}$-отказов в ***F***-модели $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ непосредственно следует, что кнопка опасна тогда и только тогда, когда она разрушающая. По правилам вывода, кнопка "P" разрушающая тогда и только тогда, когда не-отказ $\Rightarrow$ разрушающий.

3. Из утверждения 2) непосредственно следует утверждение 3).

51. Доказательство Леммы 30

По правилам вывода, $\langle\gamma\rangle\in\boldsymbol{\Sigma} \Leftrightarrow \langle\gamma\rangle\in\boldsymbol{Comp}(\boldsymbol{\Sigma})$.

Если $\langle\gamma\rangle\in\boldsymbol{\Sigma}$, то в $\boldsymbol{\Sigma}$ безопасных трасс нет.

В противном случае пустая трасса безопасна как в $\boldsymbol{\Sigma}$, так и в $\boldsymbol{Comp}(\boldsymbol{\Sigma})$.

Далее по индукции: пусть трасса $\mu\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeBy}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma})$, отказ $\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** μ и символ $\mathtt{u}\in\mathtt{P}$ или $\mathtt{u}=\mathtt{P}$ & $\mathtt{P}\in\mathfrak{R}$. Нам надо доказать:

1) $\mathtt{P}^{\rightarrow}$ ***safe by*** $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ ***after*** $\mu^{\rightarrow}$,

2) $\mu\cdot\langle\mathtt{u}\rangle\in\boldsymbol{\Sigma} \Rightarrow \mu^{\rightarrow}\cdot\langle\mathtt{u}^{\rightarrow}\rangle\in\boldsymbol{Comp}(\boldsymbol{\Sigma})$.

1. По Лемме 29, достаточно показать, что $\Rightarrow\notin\boldsymbol{G}(\mu)$.

Если отказ $\mathtt{P}$ содержится в постфиксе отказов трассы μ, то это верно.

В противном случае нужно показать, что $\Rightarrow\in\boldsymbol{safe}(\mu)$:

$\exists\lambda\in\boldsymbol{drt}(\mu)$ $\lambda\cdot\langle\Rightarrow\rangle\in\boldsymbol{tt}^{\#}(\boldsymbol{\Sigma})$, что эквивалентно:

$\exists\lambda\in\boldsymbol{drt}(\mu)$ $\lambda\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})$ & $\mathtt{P}$ ***safe by*** $\boldsymbol{\Sigma}$ ***after*** λ.

В качестве такой трассы λ можно взять саму трассу μ.

2. Нам достаточно показать, что $\mathtt{u}\in\boldsymbol{U}(\mu)$:

$\mu\cdot\langle\mathtt{u}\rangle$ согласована & $\mathtt{u}\in\boldsymbol{safe}(\mu)$ & $\mu\cdot\langle\mathtt{u}\rangle$ ***conf***.

Трасса $\mu\cdot\langle\mathtt{u}\rangle$ согласована, поскольку $\mu\cdot\langle\mathtt{u}\rangle\in\boldsymbol{\Sigma}$.

Далее $\mathtt{u}\in\boldsymbol{safe}(\mu)$, поскольку для $\lambda=\mu$ верно:

$\lambda \in \mathit{drt}(\mu)$ & $\lambda \cdot \langle u \rangle \in \mathit{tt}(\Sigma)$ и, следовательно, $\lambda \cdot \langle u \rangle \in \mathit{tt}^{\#}(\Sigma)$.

Нарушение конформности может быть только в том случае, когда:

$\exists \lambda \in \mathit{drt}(\mu \cdot \langle u \rangle)\ \exists \kappa\ \exists v\ \kappa \cdot \langle v \rangle \le \lambda$ & $\kappa \cdot \langle v \rangle \in \mathit{tt}^{\#}(\Sigma) \setminus \Sigma$.

Но это невозможно, так как $\mu \cdot \langle u \rangle \in \Sigma \Rightarrow \lambda \in \Sigma \Rightarrow \kappa \cdot \langle v \rangle \in \Sigma$.

Итак, мы показали выполнение всех условий, необходимых для того, чтобы $u \in \mathit{U}(\mu)$.

52. Доказательство Леммы 31

1. Сначала покажем, что преобразование ***Comp*** не сужает класс безопасных реализаций: $\mathit{safe}\mathfrak{I}(\Sigma) \subseteq (\mathit{safe}\mathfrak{I} \circ \mathit{Comp}(\Sigma))^{\leftarrow}$.

Допустим, это не так.

Тогда найдётся реализация $I \in \mathit{safe}\mathfrak{I}(\Sigma) \setminus ((\mathit{safe}\mathfrak{I} \circ \mathit{Comp}(\Sigma))^{\leftarrow})$.

Мы будем рассматривать реализацию $I^{\Rightarrow}$.

По Лемме 25, $I^{\Rightarrow\leftarrow} = I$.

Если $\langle \gamma \rangle \notin \mathit{Comp}(\Sigma)$, то, по правилам вывода, $\langle \gamma \rangle \notin \Sigma$. А тогда $I \in \mathit{safe}\mathfrak{I}(\Sigma)$ влечёт $\langle \gamma \rangle \notin I$ и, следовательно, $\langle \gamma \rangle \notin I^{\Rightarrow}$.

Тогда, по допущению, должна найтись такая трасса $\mu \in (\mathit{SafeBy} \circ \mathit{Comp}(\Sigma))^{\leftarrow} \cap \mathit{SafeIn}(I)$ и такой отказ $P \in \mathfrak{R} \cup \mathfrak{Q}$, что

A) $P^{\rightarrow}$ ***safe by*** $\mathit{Comp}(\Sigma)$ ***after*** $\mu^{\rightarrow}$, но

B) $P^{\rightarrow}$ ~~***safe in***~~ $I^{\Rightarrow}$ ***after*** $\mu^{\rightarrow}$, что эквивалентно P ~~***safe in***~~ I ***after*** μ.

Из A) следует $\mu^{\rightarrow} \cdot \langle ₽, \gamma \rangle \notin \mathit{Comp}(\Sigma)$.

Следовательно, a) трасса $\mu^{\rightarrow} \cdot \langle ₽ \rangle$ не согласована или b) $₽ \in \mathit{safe}(\mu)$.

Сначала рассмотрим случай a).

Поскольку трасса $\mu \in \mathit{SafeIn}(I)$, она согласована.

Следовательно, согласована трасса $\mu^{\rightarrow}$.

Поэтому несогласованность трассы $\mu^{\rightarrow}\cdot\langle ₽\rangle$ означает, что отказ $P^{\rightarrow}$ входит в постфикс отказов трассы $\mu^{\rightarrow}$, что эквивалентно: отказ P входит в постфикс отказов трассы μ.

По Лемме 28, трасса $\mu^{\rightarrow}$ не содержит $\mathfrak{Q}^{\rightarrow}$-отказов.

Следовательно, трасса μ не содержит $\mathfrak{Q}$-отказов.

Следовательно, $P\in\mathfrak{R}$.

Но это противоречит B), так как, если $\mathfrak{R}$-отказ входит в постфикс безопасной по ***safe in*** трассы, то этот $\mathfrak{R}$-отказ безопасен по ***safe in*** после этой трассы.

Остаётся случай b).

По определению $\boldsymbol{safe}(\mu)$, найдётся такая трасса $\lambda\in\boldsymbol{drt}(\mu)$, что $\lambda\cdot\langle ₽\rangle\in\boldsymbol{tt}^{\#}(\boldsymbol{\Sigma})$.

А тогда $\lambda\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})$ и P ***safe by*** $\boldsymbol{\Sigma}$ ***after*** λ.

Далее $\mu\in\boldsymbol{SafeIn}(\mathbf{I})$ влечёт $\mu\in\mathbf{I}$.

По ***drt***-замкнутости модели, $\mu\in\mathbf{I}$ влечёт $\lambda\in\mathbf{I}$.

Поскольку $\boldsymbol{drt}(\mu)\subseteq\boldsymbol{di}(\mu)$, имеем $\lambda\in\boldsymbol{di}(\mu)$.

А тогда, по Лемме 19, P ~~***safe in***~~ $\mathbf{I}$ ***after*** μ влечёт P ~~***safe in***~~ $\mathbf{I}$ ***after*** λ.

Итак, мы имеем $\lambda\in\boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\mathbf{I}$, P ***safe by*** $\boldsymbol{\Sigma}$ ***after*** λ, но P ~~***safe in***~~ $\mathbf{I}$ ***after*** λ, что противоречит по Лемме 6, $\mathbf{I}\in\boldsymbol{safe}\mathfrak{I}(\boldsymbol{\Sigma})$.

Итак, мы пришли к противоречию и, следовательно, наше допущение не верно, а утверждение о несужении класса безопасных реализаций доказано.

2. Теперь покажем, что преобразование ***Comp*** не сужает класс конформных реализаций: $\mathfrak{I}(\boldsymbol{\Sigma})\subseteq(\mathfrak{I}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma}))^{\leftarrow}$.

Допустим, это не так.

Тогда найдётся реализация $\mathbf{I}\in\mathfrak{I}(\mathbf{\Sigma})\setminus((\mathfrak{I}\circ\mathit{Comp}(\mathbf{\Sigma}))^{\leftarrow})$.

Мы будем рассматривать реализацию $\mathbf{I}^{\Rightarrow}$.

По Лемме 25, $\mathbf{I}^{\Rightarrow\leftarrow}=\mathbf{I}$.

Сначала покажем, что преобразование ***Comp*** не расширяет класс безопасных реализаций без не-отказов.

Допустим, это не так.

Тогда найдётся реализация без не-отказов:

$\mathbf{T}\in\mathit{safe}\mathfrak{I}\circ\mathit{Comp}(\mathbf{\Sigma})$ & $\mathbf{T}^{\leftarrow}\in\mathit{FMODEL}(\mathtt{L})\setminus\mathit{safe}\mathfrak{I}(\mathbf{\Sigma})$.

Если $\langle\gamma\rangle\notin\mathbf{\Sigma}$, то, по правилам вывода, $\langle\gamma\rangle\notin\mathit{Comp}(\mathbf{\Sigma})$. А тогда

$\mathbf{T}\in\mathit{safe}\mathfrak{I}\circ\mathit{Comp}(\mathbf{\Sigma})$ влечёт $\langle\gamma\rangle\notin\mathbf{T}$ и, следовательно, $\langle\gamma\rangle\notin\mathbf{T}^{\leftarrow}$.

А тогда, по допущению, должна найтись такая трасса $\mu\in\mathit{SafeBy}(\mathbf{\Sigma})\cap\mathit{SafeIn}(\mathbf{T}^{\leftarrow})$ и такой отказ $\mathtt{P}\in\mathfrak{R}\cup\mathfrak{Q}$, что

A) $\mathtt{P}$ ***safe by*** $\mathbf{\Sigma}$ ***after*** μ, но

B) $\mathtt{P}$ ~~***safe in***~~ $\mathbf{T}^{\leftarrow}$ ***after*** μ.

По Лемме 30, имеем:

$\mu^{\rightarrow}\in\mathit{SafeBy}\circ\mathit{Comp}(\mathbf{\Sigma})$ и $\mathtt{P}^{\rightarrow}$ ***safe by*** ***Comp***$(\mathbf{\Sigma})$ ***after*** $\mu^{\rightarrow}$.

Далее $\mu\in\mathit{SafeIn}(\mathbf{T}^{\leftarrow})$ влечёт $\mu^{\rightarrow}\in\mathit{SafeIn}(\mathbf{T})$.

Далее $\mathtt{P}$ ~~***safe in***~~ $\mathbf{T}^{\leftarrow}$ ***after*** μ влечёт $\mathtt{P}^{\rightarrow}$ ~~***safe in***~~ $\mathbf{T}$ ***after*** $\mu^{\rightarrow}$.

Итак: $\mu^{\rightarrow}\in\mathit{SafeBy}\circ\mathit{Comp}(\mathbf{\Sigma})\cap\mathit{SafeIn}(\mathbf{T})$

и $\mathtt{P}^{\rightarrow}$ ***safe by*** ***Comp***$(\mathbf{\Sigma})$ ***after*** $\mu^{\rightarrow}$, но $\mathtt{P}^{\rightarrow}$ ~~***safe in***~~ $\mathbf{T}$ ***after*** $\mu^{\rightarrow}$.

Однако, это противоречит $\mathbf{T}\in\mathit{safe}\mathfrak{I}\circ\mathit{Comp}(\mathbf{\Sigma})$.

Итак, мы пришли к противоречию и, следовательно, наше допущение не верно, а утверждение о нерасширении класса безопасных реализаций доказано.

По доказанному утверждению 1, реализация $\mathbf{I} \in (\boldsymbol{safe}\mathfrak{I} \circ \boldsymbol{Comp}(\mathbf{\Sigma}))^{\leftarrow}$, то есть она безопасна, и нарушено может быть только тестируемое условие конформности.

Следовательно, должна найтись такая трасса $\mu \in (\boldsymbol{SafeBy} \circ \boldsymbol{Comp}(\mathbf{\Sigma}))^{\leftarrow} \cap \boldsymbol{SafeIn}(\mathbf{I})$, такой отказ $\mathtt{P} \in \mathfrak{R} \cup \mathfrak{Q}$ и такой символ $\mathtt{u}$, что:

A) $\mathtt{P}^{\rightarrow}$ ***safe by*** $\boldsymbol{Comp}(\mathbf{\Sigma})$ ***after*** $\mu^{\rightarrow}$, но

B) $\mathtt{u} \in \boldsymbol{obs}(\mu, \mathtt{P}, \mathbf{I}) \setminus (\boldsymbol{obs}(\mu^{\rightarrow}, \mathtt{P}^{\rightarrow}, \boldsymbol{Comp}(\mathbf{\Sigma}))^{\leftarrow})$.

Покажем, что трасса $\mu^{\rightarrow} \cdot \langle \mathtt{u}^{\rightarrow} \rangle$ согласована.

Действительно, $\mathtt{u} \in \boldsymbol{obs}(\mu, \mathtt{P}, \mathbf{I})$ влечёт $\mu \cdot \langle \mathtt{u} \rangle \in \mathbf{I}$.

Следовательно, трасса $\mu \cdot \langle \mathtt{u} \rangle$ согласована.

А тогда согласована трасса $\mu^{\rightarrow} \cdot \langle \mathtt{u}^{\rightarrow} \rangle$.

Покажем, что $\mathtt{u} \in \boldsymbol{safe}(\mu)$.

Из условия A) следует $\mu^{\rightarrow} \cdot \langle ₽, \gamma \rangle \notin \boldsymbol{Comp}(\mathbf{\Sigma})$.

Следовательно, a) трасса $\mu^{\rightarrow} \cdot \langle ₽ \rangle$ не согласована или b) $₽ \in \boldsymbol{safe}(\mu)$.

Сначала рассмотрим случай a), то есть допустим, что трасса $\mu^{\rightarrow} \cdot \langle ₽ \rangle$ не согласована.

Поскольку трасса $\mu \in \boldsymbol{SafeIn}(\mathbf{I})$, она согласована.

Следовательно, согласована трасса $\mu^{\rightarrow}$.

Поэтому несогласованность трассы $\mu^{\rightarrow} \cdot \langle ₽ \rangle$ означает, что отказ $\mathtt{P}^{\rightarrow}$ входит в постфикс отказов трассы $\mu^{\rightarrow}$, что эквивалентно: отказ $\mathtt{P}$ входит в постфикс отказов трассы μ.

По Лемме 28, трасса $\mu^{\rightarrow}$ не содержит $\mathfrak{Q}^{\rightarrow}$-отказов.

Следовательно, трасса μ не содержит $\mathfrak{Q}$-отказов.

Следовательно, $P \in \mathfrak{R}$.

Поскольку $u \in \mathit{obs}(\mu, P, \mathbf{I})$, должно быть $u \in P$ или $u = P$.

Поскольку $P \in \mathit{Im} \circ \mathit{postf}(\mu)$, должно быть $u = P$.

Далее $P \in \mathit{Im} \circ \mathit{postf}(\mu)$ влечёт $P^{\rightarrow} \in \mathit{Im} \circ \mathit{postf}(\mu^{\rightarrow})$.

По свойствам модели, $P^{\rightarrow} \in \mathit{Im} \circ \mathit{postf}(\mu^{\rightarrow})$ влечёт $\mu^{\rightarrow} \cdot \langle P^{\rightarrow} \rangle \in \mathit{Comp}(\mathbf{\Sigma})$.

Но для $u = P$ это противоречит $u \notin \mathit{obs}(\mu^{\rightarrow}, P^{\rightarrow}, \mathit{Comp}(\mathbf{\Sigma}))^{\leftarrow}$.

Мы пришли к противоречию и, следовательно, случай a) не имеет места. Следовательно, остаётся случай b): $\not\Vdash \in \mathit{safe}(\mu)$.

По определению $\mathit{safe}(\mu)$ и $\mathit{tt}^{\#}(\mathbf{\Sigma})$, $\not\Vdash \in \mathit{safe}(\mu)$ влечёт:

$\exists \lambda \in \mathit{drt}(\mu)$ $\lambda \in \mathit{SafeBy}(\mathbf{\Sigma})$ & P ***safe by*** $\mathbf{\Sigma}$ ***after*** λ.

Поскольку $u \in P$ или $u = P$, имеем $u \in \mathit{safe}(\mu)$.

Теперь покажем, что $\mu \cdot \langle u \rangle$ ***conf***.

Допустим, это не так:

$\exists \lambda \in \mathit{drt}(\mu \cdot \langle u \rangle)$ $\exists \kappa$ $\exists v$ $\kappa \cdot \langle v \rangle \leq \lambda$ & $\kappa \cdot \langle v \rangle \in \mathit{tt}^{\#}(\mathbf{\Sigma}) \setminus \mathbf{\Sigma}$.

Но, если $\mu \cdot \langle u \rangle \in \mathbf{I}$, то, по ***drt***-замкнутости модели $\mathbf{I}$, имеем $\lambda \in \mathbf{I}$.

Следовательно, $\kappa \cdot \langle v \rangle \in \mathbf{I}$ и $\kappa \in \mathbf{I}$.

Из $\kappa \cdot \langle v \rangle \in \mathit{tt}^{\#}(\mathbf{\Sigma}) \setminus \mathbf{\Sigma}$ следует $\kappa \in \mathit{SafeBy}(\mathbf{\Sigma})$ и существует отказ $Q \in \mathfrak{R} \cup \mathfrak{Q}$ такой, что Q ***safe by*** $\mathbf{\Sigma}$ ***after*** κ, также $v = Q$ & $Q \in \mathfrak{R}$ или $v \in Q$, но $v \notin \mathit{obs}(\kappa, Q, \mathbf{\Sigma})$.

Имеем: $\kappa \in \mathit{SafeBy}(\mathbf{\Sigma}) \cap \mathbf{I}$ & $Q \in \mathfrak{R} \cup \mathfrak{Q}$ & Q ***safe by*** $\mathbf{\Sigma}$ ***after*** κ & $\mathit{obs}(\kappa, Q, \mathbf{I}) \not\subseteq \mathit{obs}(\kappa, Q, \mathbf{\Sigma})$, что противоречит $\mathbf{I} \in \mathfrak{I}(\mathbf{\Sigma})$.

Мы пришли к противоречию и, следовательно, наше допущение было не верно, и $\mu \cdot \langle u \rangle$ ***conf***.

Итак, мы доказали: $\mu\cdot\langle u\rangle$ согласована & $u\in \boldsymbol{safe}(\mu)$ & $\mu\cdot\langle u\rangle$ ***conf***.

Следовательно, $u\in \boldsymbol{U}(\mu)$.

Тогда из $\mu^{\rightarrow}\in \boldsymbol{Comp}(\boldsymbol{\Sigma})$ и правил вывода следует $\mu^{\rightarrow}\cdot\langle u^{\rightarrow}\rangle\in \boldsymbol{Comp}(\boldsymbol{\Sigma})$.

А тогда $u\in \boldsymbol{obs}(\mu^{\rightarrow}, P^{\rightarrow}, \boldsymbol{Comp}(\boldsymbol{\Sigma}))^{\leftarrow}$, что противоречит условию B).

Итак, мы пришли к противоречию и, следовательно, наше допущение не верно, а утверждение о несужении класса конформных реализаций доказано.

53. Доказательство Леммы 32

1. Сначала покажем, что преобразование ***Comp*** не расширяет класс безопасных реализаций без не-отказов.

Допустим, это не так.

Тогда найдётся реализация без не-отказов:

$\mathbf{I}\in \boldsymbol{safe\mathfrak{I}}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma})$ & $\mathbf{I}^{\leftarrow}\in \boldsymbol{FMODEL}(L)\setminus \boldsymbol{safe\mathfrak{I}}(\boldsymbol{\Sigma})$.

Если $\langle\gamma\rangle\notin\boldsymbol{\Sigma}$, то, по правилам вывода, $\langle\gamma\rangle\notin \boldsymbol{Comp}(\boldsymbol{\Sigma})$. А тогда

$\mathbf{I}\in \boldsymbol{safe\mathfrak{I}}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma})$ влечёт $\langle\gamma\rangle\notin\mathbf{I}$ и, следовательно, $\langle\gamma\rangle\notin\mathbf{I}^{\leftarrow}$.

А тогда, по допущению, должна найтись такая трасса $\mu\in \boldsymbol{SafeBy}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I}^{\leftarrow})$ и такой отказ $P\in\mathfrak{R}\cup\mathfrak{Q}$, что

A) P ***safe by*** $\boldsymbol{\Sigma}$ ***after*** μ, но

B) P ~~***safe in***~~ $\mathbf{I}^{\leftarrow}$ ***after*** μ.

По Лемме 30, имеем:

$\mu^{\rightarrow}\in \boldsymbol{SafeBy}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma})$ и $P^{\rightarrow}$ ***safe by*** $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ ***after*** $\mu^{\rightarrow}$.

Далее $\mu\in \boldsymbol{SafeIn}(\mathbf{I}^{\leftarrow})$ влечёт $\mu^{\rightarrow}\in \boldsymbol{SafeIn}(\mathbf{I})$.

Далее P ~~***safe in***~~ $\mathbf{I}^{\leftarrow}$ ***after*** μ влечёт $P^{\rightarrow}$ ~~***safe in***~~ $\mathbf{I}$ ***after*** $\mu^{\rightarrow}$.

Итак:

$\mu^{\rightarrow}\in \boldsymbol{SafeBy}\circ\boldsymbol{Comp}(\boldsymbol{\Sigma})\cap\boldsymbol{SafeIn}(\mathbf{I})$ и $P^{\rightarrow}$ ***safe by*** $\boldsymbol{Comp}(\boldsymbol{\Sigma})$ ***after*** $\mu^{\rightarrow}$, но $P^{\rightarrow}$ ~~***safe in***~~ $\mathbf{I}$ ***after*** $\mu^{\rightarrow}$.

Однако, это противоречит $\mathbf{I}\in\boldsymbol{safe}\mathfrak{I}\circ\boldsymbol{Comp}(\Sigma)$.

Итак, мы пришли к противоречию и, следовательно, наше допущение не верно, а утверждение о нерасширении класса безопасных реализаций доказано.

2. Теперь покажем, что преобразование ***Comp*** не расширяет класс конформных реализаций без не-отказов.

Допустим, это не так.

Тогда найдётся реализация без не-отказов:

$\mathbf{I}\in\mathfrak{I}\circ\boldsymbol{Comp}(\Sigma)\ \ \&\ \ \mathbf{I}^{\leftarrow}\in\boldsymbol{FMODEL}(L)\setminus\mathfrak{I}(\Sigma)$.

По доказанному утверждению 1, реализация $\mathbf{I}^{\leftarrow}\in\boldsymbol{safe}\mathfrak{I}(\Sigma)$, то есть она безопасна и нарушено может быть только тестируемое условие конформности.

Следовательно, должна найтись такая трасса $\mu\in\boldsymbol{SafeBy}(\Sigma)\cap\mathbf{I}^{\leftarrow}$, такой отказ $P\in\mathfrak{R}\cup\mathfrak{Q}$ и такой символ u, что:

A) P ***safe by*** Σ ***after*** μ, но

B) $u\in\boldsymbol{obs}(\mu,P,\mathbf{I}^{\leftarrow})\setminus\boldsymbol{obs}(\mu,P,\Sigma)$.

По Лемме 30, имеем:

$\mu^{\rightarrow}\in\boldsymbol{SafeBy}\circ\boldsymbol{Comp}(\Sigma)$ и $P^{\rightarrow}$ ***safe by*** $\boldsymbol{Comp}(\Sigma)$ ***after*** $\mu^{\rightarrow}$.

Далее $\mu\in\mathbf{I}^{\leftarrow}$ влечёт $\mu^{\rightarrow}\in\mathbf{I}$.

Далее $u\in\boldsymbol{obs}(\mu,P,\mathbf{I}^{\leftarrow})$ влечёт $u^{\rightarrow}\in\boldsymbol{obs}(\mu,P,\mathbf{I})$.

Итак: $\mu^{\rightarrow}\in\boldsymbol{SafeBy}\circ\boldsymbol{Comp}(\Sigma)\cap\mathbf{I}$, $P^{\rightarrow}$ ***safe by*** $\boldsymbol{Comp}(\Sigma)$ ***after*** $\mu^{\rightarrow}$ и $u^{\rightarrow}\in\boldsymbol{obs}(\mu,P,\mathbf{I})$.

Тогда, поскольку $\mathbf{I}\in\mathfrak{I}\circ\boldsymbol{Comp}(\Sigma)$, имеем $u^{\rightarrow}\in\boldsymbol{obs}(\mu^{\rightarrow},P^{\rightarrow},\boldsymbol{Comp}(\Sigma))$.

Поскольку трассы реализации $\mathbf{I}$ не имеют не-отказов, а $u^{\rightarrow}\in\boldsymbol{obs}(\mu,P,\mathbf{I})$, символ u является действием из алфавита L или $\mathfrak{R}$-отказом (то есть не не-отказом).

Следовательно, по правилам вывода, $u \in \boldsymbol{U}(\mu)$, что влечёт $\mu \cdot \langle u \rangle$ ***conf***.

Поскольку P ***safe by*** $\boldsymbol{\Sigma}$ ***after*** μ и $u \in \boldsymbol{obs}(\mu, P, \mathbf{I}^{\leftarrow})$, имеем $\mu \cdot \langle u \rangle \in \boldsymbol{tt}^{\#}(\boldsymbol{\Sigma})$.

А тогда, по определению ***conf***, должно быть $\mu \cdot \langle u \rangle \in \boldsymbol{\Sigma}$.

Но это противоречит $u \notin \boldsymbol{obs}(\mu, P, \boldsymbol{\Sigma})$.

Итак, мы пришли к противоречию и, следовательно, наше допущение не верно, а утверждение о нерасширении класса конформных реализаций доказано.

54. **Доказательство Леммы 33**

По Лемме 29, отношение ***safe by*** одинаково определяется для ***F***-модели ***Comp***$(\boldsymbol{\Sigma})$ как в $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$-, так и в $\mathfrak{R}^{\rightarrow} \cup \mathfrak{Q}^{\rightarrow}/\varnothing$-семантике. Поэтому требования к безопасной реализации совпадают, за исключением того, что в $\mathfrak{R}^{\rightarrow}/\mathfrak{Q}^{\rightarrow}$-семантике после трассы, безопасной в спецификации, в реализации не должно быть $\mathfrak{Q}^{\rightarrow}$-отказа, если он безопасен в спецификации. Отсюда непосредственно следуют оба утверждения Леммы.

55. **Доказательство Теоремы 22**

Утверждение Теоремы непосредственно следует из Леммы 31, Леммы 32 и Леммы 33.

56. **Доказательство Теоремы 23**

Пусть заданы алфавиты $L, L_1, L_2, L_3 \subseteq Z^{\#}$, причём $L_3 = L_1 \upharpoonright\!\downarrow L_2$, и $\mathfrak{R}$-, $\mathfrak{R}_1$-, $\mathfrak{R}_2$, $\mathfrak{R}_3$-семантики такие, что $\cup\mathfrak{R} = L$, $\cup\mathfrak{R}_1 = L_1$, $\cup\mathfrak{R}_2 = L_2$, $\cup\mathfrak{R}_3 = L_3$, а также спецификации $\mathbf{S} \in \mathfrak{C}(L_\gamma)$, $\mathbf{S}_1 \in \mathfrak{C}(L_{1\gamma})$, $\mathbf{S}_2 \in \mathfrak{C}(L_{2\gamma})$ и $\mathbf{S}_3 \in \mathfrak{C}(L_{3\gamma})$.

Нам нужно доказать следующие утверждения при условии выполнения условий монотонности 1÷7:

1) Инвариантность преобразований: $\mathcal{T}_{\mathfrak{R}}(\mathbf{S}) \sim_{\mathfrak{R}} \mathbf{S}$.

2) Композиция преобразованных спецификаций является корректной спецификацией:

$\forall \mathbf{I}_1 \in \mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1) \quad \forall \mathbf{I}_2 \in \mathfrak{I}_{\mathfrak{R}2}(\mathbf{S}_2) \quad \mathbf{I}_1 \upharpoonright\downarrow \mathbf{I}_2 \quad \mathit{saco}_{\mathfrak{R}3} \quad \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \upharpoonright\downarrow \mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2)$.

3) Композиция преобразованных спецификаций является левокорректной спецификацией при выполнения дополнительного условия Монотонность 8::

$\forall \mathbf{I}_1 \in \mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1) \quad \forall \mathbf{I}_2 \in \mathfrak{C}(\mathrm{L}_{2\gamma}) \quad \mathbf{I}_1 \upharpoonright\downarrow \mathbf{I}_2 \quad \mathit{saco}_{\mathfrak{R}3} \quad \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \upharpoonright\downarrow \mathbf{I}_2$.

4) Композиция преобразованных спецификаций является самой сильной корректной спецификацией:

$(\forall \mathbf{I}_1 \in \mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1) \quad \forall \mathbf{I}_2 \in \mathfrak{I}_{\mathfrak{R}2}(\mathbf{S}_2) \quad \mathbf{I}_1 \upharpoonright\downarrow \mathbf{I}_2 \quad \mathit{saco}_{\mathfrak{R}3} \quad \mathbf{S}_3)$

$\Rightarrow \quad \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \upharpoonright\downarrow \mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2) \quad \mathit{saco}_{\mathfrak{R}3} \quad \mathbf{S}_3$.

5) Композиция преобразованной спецификации является самой сильной левокорректной спецификацией при выполнения дополнительного условия Монотонность 8: $\forall \mathbf{I}_2 \in \mathfrak{C}(\mathrm{L}_{2\gamma})$:

$(\forall \mathbf{I}_1 \in \mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1) \quad \mathbf{I}_1 \upharpoonright\downarrow \mathbf{I}_2 \quad \mathit{saco}_{\mathfrak{R}3} \quad \mathbf{S}_3) \Rightarrow \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \upharpoonright\downarrow \mathbf{I}_2 \quad \mathit{saco}_{\mathfrak{R}3} \quad \mathbf{S}_3$.

Доказательство утверждения 1).

По конформности преобразований (Монотонность 6:),

$\mathcal{T}_{\mathfrak{R}}(\mathbf{S}) \quad \mathit{saco}_{\mathfrak{R}} \quad \mathbf{S}$.

Докажем обратное: $\mathbf{S} \quad \mathit{saco}_{\mathfrak{R}} \quad \mathcal{T}_{\mathfrak{R}}(\mathbf{S})$.

По рефлексивности предпорядка,

$\mathbf{S} \quad \mathit{saco}_{\mathfrak{R}} \quad \mathbf{S}$.

Тогда, по мажорантности преобразований (Монотонность 7:),

$\mathbf{\Phi}^{\omega}(\mathbf{S}) \quad \preccurlyeq \quad \mathbf{\Phi}^{\omega} \circ \mathcal{T}_{\mathfrak{R}}(\mathbf{S})$.

Тогда, по генеративности ϕ-мажорирования (Монотонность 4:),

$\cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}(\mathbf{S}) \preccurlyeq_{\mathfrak{R}} \cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}\circ\mathcal{T}_{\mathfrak{R}}(\mathbf{S})$.

Тогда, по генеративности ϕ-трасс (Монотонность 2:),

$\boldsymbol{F}(\mathbf{S}) \preccurlyeq_{\mathfrak{R}} \boldsymbol{F}\circ\mathcal{T}_{\mathfrak{R}}(\mathbf{S})$.

Тогда, поскольку $\mathfrak{R}$-мажорирование влечёт конформность (Монотонность 1:),

$\mathbf{S}\ \boldsymbol{saco}_{\mathfrak{R}}\ \mathcal{T}_{\mathfrak{R}}(\mathbf{S})$.

Утверждение 1) доказано.

Доказательство утверждения 2).

Из $\mathbf{I}_1\in\mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1)$ и $\mathbf{I}_2\in\mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_2)$ следует, по мажорантности преобразований (Монотонность 7:),

$\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1) \preccurlyeq \boldsymbol{\Phi}^{\omega}\circ\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1)$ и $\boldsymbol{\Phi}^{\omega}(\mathbf{I}_2) \preccurlyeq \boldsymbol{\Phi}^{\omega}\circ\mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2)$.

Тогда, по композиционности ϕ-мажорирования (Монотонность 5:),

$\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1)\wr\boldsymbol{\Phi}^{\omega}(\mathbf{I}_2)) \preccurlyeq \cup(\boldsymbol{\Phi}^{\omega}\circ\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1)\wr\boldsymbol{\Phi}^{\omega}\circ\mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2))$.

Тогда, по аддитивности ϕ-трасс (Монотонность 3:),

$\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1\wr\mathbf{I}_2) \preccurlyeq \boldsymbol{\Phi}^{\omega}(\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1)\wr\mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2))$.

Тогда, по генеративности ϕ-мажорирования (Монотонность 4:),

$\cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1\wr\mathbf{I}_2) \preccurlyeq_{\mathfrak{R}3} \cup\circ\xi\circ\boldsymbol{\Phi}^{\omega}(\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1)\wr\mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2))$.

Тогда, по генеративности ϕ-трасс (Монотонность 2:),

$\boldsymbol{F}(\mathbf{I}_1\wr\mathbf{I}_2) \preccurlyeq_{\mathfrak{R}3} \boldsymbol{F}(\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1)\wr\mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2))$.

Тогда, поскольку $\mathfrak{R}$-мажорирование влечёт конформность (Монотонность 1:),

$\mathbf{I}_1 \wr\!\mid\! \mathbf{I}_2 \;\; saco_{\mathfrak{R}3} \;\; \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2)$.

Утверждение 2) доказано.

Доказательство утверждения 3).

Из $\mathbf{I}_1 \in \mathfrak{I}_{\mathfrak{R}1}(\mathbf{S}_1)$ следует, по мажорантности преобразований (Монотонность 7:),

$\mathbf{\Phi}^{\omega}(\mathbf{I}_1) \preccurlyeq \mathbf{\Phi}^{\omega} \circ \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1)$.

По рефлексивности ϕ-мажорирования (Монотонность 8:),

$\mathbf{\Phi}^{\omega}(\mathbf{I}_2) \preccurlyeq \mathbf{\Phi}^{\omega}(\mathbf{I}_2)$.

Тогда, по композиционности ϕ-мажорирования (Монотонность 5:),

$\cup(\mathbf{\Phi}^{\omega}(\mathbf{I}_1) \wr\!\mid\! \mathbf{\Phi}^{\omega}(\mathbf{I}_2)) \preccurlyeq \cup(\mathbf{\Phi}^{\omega} \circ \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathbf{\Phi}^{\omega}(\mathbf{I}_2))$.

Тогда, по аддитивности ϕ-трасс (Монотонность 3:),

$\mathbf{\Phi}^{\omega}(\mathbf{I}_1 \wr\!\mid\! \mathbf{I}_2) \preccurlyeq \mathbf{\Phi}^{\omega}(\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathbf{I}_2)$.

Тогда, по генеративности ϕ-мажорирования (Монотонность 4:),

$\cup \circ \xi \circ \mathbf{\Phi}^{\omega}(\mathbf{I}_1 \wr\!\mid\! \mathbf{I}_2) \preccurlyeq_{\mathfrak{R}3} \cup \circ \xi \circ \mathbf{\Phi}^{\omega}(\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathbf{I}_2)$.

Тогда, по генеративности ϕ-трасс (Монотонность 2:),

$\boldsymbol{F}(\mathbf{I}_1 \wr\!\mid\! \mathbf{I}_2) \preccurlyeq_{\mathfrak{R}3} \boldsymbol{F}(\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathbf{I}_2)$.

Тогда, поскольку $\mathfrak{R}$-мажорирование влечёт конформность (Монотонность 1:),

$\mathbf{I}_1 \wr\!\mid\! \mathbf{I}_2 \;\; saco_{\mathfrak{R}3} \;\; \mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathbf{I}_2$.

Утверждение 3) доказано.

Доказательство утверждения 4).

По утверждению 2) композиция $\mathcal{T}_{\mathfrak{R}1}(\mathbf{S}_1) \wr\!\mid\! \mathcal{T}_{\mathfrak{R}2}(\mathbf{S}_2)$ является корректной спецификацией.

По конформности преобразований (Монотонность 6:),

$\mathcal{T}_{\mathfrak{R}1}(\mathbf{S_1})$ $saco_{\mathfrak{R}1}$ $\mathbf{S_1}$ и $\mathcal{T}_{\mathfrak{R}2}(\mathbf{S_2})$ $saco_{\mathfrak{R}2}$ $\mathbf{S_2}$.

Поэтому для корректной спецификации $\mathbf{S_3}$ имеем

$\mathcal{T}_{\mathfrak{R}1}(\mathbf{S_1}) \upharpoonright\!\downarrow \mathcal{T}_{\mathfrak{R}2}(\mathbf{S_2})$ $saco_{\mathfrak{R}3}$ $\mathbf{S_3}$.

Утверждение 4) доказано.

Доказательство утверждения 5).

По утверждению 3) композиция $\mathcal{T}_{\mathfrak{R}1}(\mathbf{S_1}) \upharpoonright\!\downarrow \mathbf{I_2}$ является левокорректной спецификацией.

По конформности преобразований (Монотонность 6:),

$\mathcal{T}_{\mathfrak{R}1}(\mathbf{S_1})$ $saco_{\mathfrak{R}1}$ $\mathbf{S_1}$.

Поэтому для левокорректной спецификации $\mathbf{S_3}$ имеем

$\mathcal{T}_{\mathfrak{R}1}(\mathbf{S_1}) \upharpoonright\!\downarrow \mathbf{I_2}$ $saco_{\mathfrak{R}3}$ $\mathbf{S_3}$.

Утверждение 5) доказано.

Основное утверждение доказано.

57. Доказательство Теоремы 24

Обозначим: $\mathbf{I}=\boldsymbol{F}(\mathbf{I})$, $\mathbf{\Sigma}=\boldsymbol{F}(\mathbf{S})$.

По определению, $\mathbf{I} \preccurlyeq_{\mathfrak{R}} \mathbf{\Sigma} \Leftrightarrow \forall \mu \in \mathbf{I}\ \mu \preccurlyeq_{\mathfrak{R}} \mathbf{\Sigma}$.

Поскольку $\mathfrak{R}$-мажорирование $\boldsymbol{F}$-трасс означает $\mathfrak{R}$-мажорирование их $\mathfrak{R}$-проекций, нам достаточно ограничиться случаем, когда μ – это $\mathfrak{R}$-трасса.

1. Сначала докажем обратную импликацию

$\forall \mu \in \mathbf{I}^{\downarrow \mathrm{L}_{\mathfrak{R}\Delta\gamma}}\ \mu \preccurlyeq_{\mathfrak{R}} \mathbf{\Sigma} \Leftarrow \mathbf{I}\ saco_{\mathfrak{R}}\ \mathbf{\Sigma}$.

Возможны два случая в зависимости от того, безопасна или опасна в спецификации $\mathbf{\Sigma}$ $\mathfrak{R}$-трасса μ.

1.1. $\mu \in \mathit{Safe}(\Sigma)$.

Тогда $\mu \in \Sigma$, следовательно, $\mu \preccurlyeq_{\mathfrak{R}} \Sigma$.

1.2. $\mu \notin \mathit{Safe}(\Sigma)$.

Тогда возможны два случая в зависимости от наличия или отсутствия в Σ безопасных $\mathfrak{R}$-трасс.

1.2.1. $\langle\gamma\rangle \in \Sigma$, безопасных $\mathfrak{R}$-трасс нет.

Тогда $\mu \preccurlyeq_{\mathfrak{R}} \Sigma$.

1.2.2. $\langle\gamma\rangle \notin \Sigma$, пустая $\mathfrak{R}$-трасса безопасна.

Тогда у $\mathfrak{R}$-трассы μ существует максимальный безопасный префикс: $\mu = \kappa \cdot \langle u \rangle \cdot \lambda$, где $\kappa \in \mathit{Safe}(\Sigma)$, $u \in \mathtt{L}_{\mathfrak{R}\Delta\gamma}$ и $\kappa \cdot \langle u \rangle \notin \mathit{Safe}(\Sigma)$.

Возможны два случая в зависимости от того, является ли символ u дивергенцией или нет.

1.2.2.1. $u = \Delta$.

Тогда, поскольку в $\mathfrak{R}$-трассе дивергенция может быть только последним символом, $\mu = \kappa \cdot \langle\Delta\rangle$, и все $\mathfrak{R}$-отказы опасны в реализации $\mathbf{I}$ после трассы κ.

А тогда, по гипотезе о безопасности для конформной реализации $\mathbf{I}$, все $\mathfrak{R}$-отказы должны быть опасными в Σ после κ.

Это может быть в двух случаях.

1.2.2.1.1. $\kappa \cdot \langle\Delta\rangle \in \Sigma$.

Тогда, поскольку $\mu = \kappa \cdot \langle\Delta\rangle \in \Sigma$, имеем $\mu \preccurlyeq_{\mathfrak{R}} \Sigma$.

1.2.2.1.2. $\forall P \in \mathfrak{R}\ \exists t \in P\ \kappa \cdot \langle t, \gamma \rangle \in \Sigma$.

Тогда, поскольку $\mu=\kappa\cdot\langle\Delta\rangle$, имеем $\mu\preccurlyeq_{\mathfrak{R}}^{\Delta\gamma}\boldsymbol{\Sigma}$.

1.2.2.2. $u\neq\Delta$.

Символ $u\neq\gamma$, так как иначе $\mathfrak{R}$-трасса κ не была бы безопасной.

Следовательно, символ u – это опасное внешнее действие или опасный $\mathfrak{R}$-отказ: u ***safe*** $\boldsymbol{\Sigma}$ ***after*** κ.

1.2.2.2.1. $u=z\in L$.

Тогда $\forall P\in\mathfrak{R}\ (z\in P \Rightarrow \exists t\in P\ \kappa\cdot\langle t,\gamma\rangle\in\boldsymbol{\Sigma})$.

А тогда $\mu\preccurlyeq_{\mathfrak{R}}^{z}\boldsymbol{\Sigma}$.

1.2.2.2.2. $u=P\in\mathfrak{R}$.

Тогда $\exists z\in P\ \pi\cdot\langle z,\gamma\rangle\in\boldsymbol{\Sigma}$.

А тогда $\mu\preccurlyeq_{\mathfrak{R}}^{P}\boldsymbol{\Sigma}$.

Первое утверждение доказано.

2. Теперь докажем прямую импликацию

$\forall\mu\in\mathbf{I}^{\downarrow}L_{\mathfrak{R}\Delta\gamma}\ \mu\preccurlyeq_{\mathfrak{R}}\boldsymbol{\Sigma} \Rightarrow \mathbf{I}\ \boldsymbol{saco}_{\mathfrak{R}}\ \boldsymbol{\Sigma}$.

2.1. Сначала докажем выполнение гипотезы о безопасности: $\mathbf{I}$ ***safe for*** $\boldsymbol{\Sigma}$.

Нам нужно доказать выполнение двух условий.

2.1.1. $\langle\gamma\rangle\notin\boldsymbol{\Sigma} \Rightarrow \langle\gamma\rangle\notin\mathbf{I}$.

Действительно, если бы $\langle\gamma\rangle\in\mathbf{I}$, то должно было бы быть $\langle\gamma\rangle\preccurlyeq_{\mathfrak{R}}\boldsymbol{\Sigma}$, что возможно только в том случае, когда $\langle\gamma\rangle\in\boldsymbol{\Sigma}$.

2.1.2. По Лемме 6, должно быть: $\forall\mu\in\boldsymbol{Safe}(\boldsymbol{\Sigma})\cap\mathbf{I}\ \forall P\in\mathfrak{R}$

$(P$ ***safe*** $\boldsymbol{\Sigma}$ ***after*** $\mu \Rightarrow P$ ***safe*** $\mathbf{I}$ ***after*** $\mu)$.

Допустим, это не так: P ~~***safe***~~ $\mathbf{I}$ ***after*** μ.

Тогда возможны два случая.

2.1.2.1. $\mu\cdot\langle\Delta\rangle\in\mathbf{I}$.

Поскольку $\mu\cdot\langle\Delta\rangle\preccurlyeq_{\mathfrak{R}}\boldsymbol{\Sigma}$, должно выполняться одно из шести условий.

2.1.2.1.1. $\mu\cdot\langle\Delta\rangle\in\boldsymbol{\Sigma}$. Но это противоречит $\mathtt{P}$ ***safe*** $\boldsymbol{\Sigma}$ ***after*** μ.

2.1.2.1.2. $\langle\gamma\rangle\in\boldsymbol{\Sigma}$, что противоречит $\mu\in\boldsymbol{Safe}(\boldsymbol{\Sigma})$.

2.1.2.1.3. $\mu\cdot\langle\Delta\rangle\preccurlyeq^{\Delta}_{\mathfrak{R}}\boldsymbol{\Sigma} = \exists\pi\ \pi\le\mu\cdot\langle\Delta\rangle\ \&\ \mu\cdot\langle\Delta\rangle\ne\pi\cdot\langle\gamma\rangle\ \&\ \pi\cdot\langle\Delta\rangle\in\boldsymbol{\Sigma}$. Поскольку трассы $\mu\cdot\langle\Delta\rangle\cdot\langle\Delta\rangle$ не может быть, должно быть $\pi\le\mu$ и $\pi\cdot\langle\Delta\rangle\in\boldsymbol{\Sigma}$. Для $\pi=\mu$ это противоречит $\mathtt{P}$ ***safe*** $\boldsymbol{\Sigma}$ ***after*** μ, а для $\pi<\mu$ это противоречит $\mu\in\boldsymbol{Safe}(\boldsymbol{\Sigma})$.

2.1.2.1.4. $\mu\cdot\langle\Delta\rangle\preccurlyeq^{\Delta\gamma}_{\mathfrak{R}}\boldsymbol{\Sigma} = \exists\pi\ \mu\cdot\langle\Delta\rangle=\pi\cdot\langle\Delta\rangle\ \&\ \forall\mathtt{R}\in\mathfrak{R}\ \exists\mathtt{t}\in\mathtt{R}\ \pi\cdot\langle\mathtt{t},\gamma\rangle\in\boldsymbol{\Sigma}$. Это влечёт $\forall\mathtt{R}\in\mathfrak{R}\ \exists\mathtt{t}\in\mathtt{R}\ \mu\cdot\langle\mathtt{t},\gamma\rangle\in\boldsymbol{\Sigma}$, что противоречит $\mathtt{P}$ ***safe*** $\boldsymbol{\Sigma}$ ***after*** μ.

2.1.2.1.5. $\mu\cdot\langle\Delta\rangle\preccurlyeq^{\mathtt{P}}_{\mathfrak{R}}\boldsymbol{\Sigma} = \exists\pi\ \exists\mathtt{R}\in\mathfrak{R}\ \pi\cdot\langle\mathtt{R}\rangle\le\mu\cdot\langle\Delta\rangle\ \&\ \exists\mathtt{z}\in\mathtt{R}\ \pi\cdot\langle\mathtt{z},\gamma\rangle\in\boldsymbol{\Sigma}$.

Поскольку $\mathtt{R}\ne\Delta$, это влечёт $\pi\cdot\langle\mathtt{R}\rangle\le\mu\ \&\ \exists\mathtt{z}\in\mathtt{R}\ \pi\cdot\langle\mathtt{z},\gamma\rangle\in\boldsymbol{\Sigma}$, что влечёт $\mathtt{R}$ ~~***safe***~~ $\boldsymbol{\Sigma}$ ***after*** π, что противоречит $\mu\in\boldsymbol{Safe}(\boldsymbol{\Sigma})$.

2.1.2.1.6. $\mu\cdot\langle\Delta\rangle\preccurlyeq^{\mathtt{z}}_{\mathfrak{R}}\boldsymbol{\Sigma} = \exists\pi\ \exists\mathtt{z}\in\mathtt{L}\ \pi\cdot\langle\mathtt{z}\rangle\le\mu\cdot\langle\Delta\rangle\ \&\ \forall\mathtt{R}\in\mathfrak{R}\ (\mathtt{z}\in\mathtt{R}\ \Rightarrow\ \exists\mathtt{t}\in\mathtt{R}\ \pi\cdot\langle\mathtt{t},\gamma\rangle\in\boldsymbol{\Sigma})$. Поскольку $\mathtt{z}\ne\Delta$, это влечёт: $\pi\cdot\langle\mathtt{z}\rangle\le\mu\ \&\ \forall\mathtt{R}\in\mathfrak{R}\ (\mathtt{z}\in\mathtt{R}\ \Rightarrow\ \exists\mathtt{t}\in\mathtt{R}\ \pi\cdot\langle\mathtt{t},\gamma\rangle\in\boldsymbol{\Sigma})$, что влечёт $\mathtt{z}$ ~~***safe***~~ $\boldsymbol{\Sigma}$ ***after*** π, что противоречит $\mu\in\boldsymbol{Safe}(\boldsymbol{\Sigma})$.

Мы пришли к противоречию во всех шести случаях, следовательно, не должно быть $\mu\cdot\langle\Delta\rangle\in\mathbf{I}$.

2.1.2.2. $\mu\cdot\langle\Delta\rangle\notin\mathbf{I}$, но $\exists z\in P\ \mu\cdot\langle z,\gamma\rangle\in\mathbf{I}$.

Поскольку $\mu\cdot\langle z,\gamma\rangle\preccurlyeq_{\mathfrak{R}}\mathbf{\Sigma}$, должно выполняться одно из шести условий.

2.1.2.2.1. $\mu\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}$. Но это противоречит P ***safe*** $\mathbf{\Sigma}$ ***after*** μ.

2.1.2.2.2. $\langle\gamma\rangle\in\mathbf{\Sigma}$, что противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

2.1.2.2.3. $\mu\cdot\langle z,\gamma\rangle\preccurlyeq_{\mathfrak{R}}^{\Delta}\mathbf{\Sigma}=\exists\pi\ \pi\le\mu\cdot\langle z,\gamma\rangle\ \&\ \mu\cdot\langle z,\gamma\rangle\neq\pi\cdot\langle\gamma\rangle\ \&\ \pi\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$. Тогда должно быть $\pi\le\mu$ и $\pi\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$. Для $\pi=\mu$ это противоречит P ***safe*** $\mathbf{\Sigma}$ ***after*** μ, а для $\pi<\mu$ это противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

2.1.2.2.4. $\mu\cdot\langle z,\gamma\rangle\preccurlyeq_{\mathfrak{R}}^{\Delta\gamma}\mathbf{\Sigma}=\exists\pi\ \mu\cdot\langle z,\gamma\rangle=\pi\cdot\langle\Delta\rangle\ \&\ \forall R\in\mathfrak{R}\ \exists t\in R$ $\pi\cdot\langle t,\gamma\rangle\in\mathbf{\Sigma}$. Но этого не может быть, так как $\gamma\neq\Delta$.

2.1.2.2.5. $\mu\cdot\langle z,\gamma\rangle\preccurlyeq_{\mathfrak{R}}^{P}\mathbf{\Sigma}=\exists\pi\ \exists R\in\mathfrak{R}\ \pi\cdot\langle R\rangle\le\mu\cdot\langle z,\gamma\rangle\ \&\ \exists t\in R\ \pi\cdot\langle t,\gamma\rangle\in\mathbf{\Sigma}$.

Поскольку $R\neq\Delta$ и $R\neq\gamma$, это влечёт $\pi\cdot\langle R\rangle\le\mu\ \&\ \exists t\in R\ \pi\cdot\langle t,\gamma\rangle\in\mathbf{\Sigma}$, что влечёт R ***safe*** $\mathbf{\Sigma}$ ***after*** π, что противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

2.1.2.2.6. $\mu\cdot\langle z,\gamma\rangle\preccurlyeq_{\mathfrak{R}}^{z}\mathbf{\Sigma}=\exists\pi\ \exists v\in L\ \pi\cdot\langle v\rangle\le\mu\cdot\langle z,\gamma\rangle\ \&\ \forall R\in\mathfrak{R}\ (v\in R\Rightarrow\exists t\in R\ \pi\cdot\langle t,\gamma\rangle\in\mathbf{\Sigma})$. Поскольку $v\neq\gamma$, это влечёт: $\pi\cdot\langle v\rangle\le\mu\cdot\langle z\rangle\ \&\ \forall R\in\mathfrak{R}\ (v\in R\Rightarrow\exists t\in R\ \pi\cdot\langle t,\gamma\rangle\in\mathbf{\Sigma})$. Для $\pi=\mu$ будет $v=z$, что противоречит P ***safe*** $\mathbf{\Sigma}$ ***after*** μ, а для $\pi<\mu$ это противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

Мы пришли к противоречию во всех шести случаях, следовательно, не должно быть $\mu\cdot\langle\Delta\rangle\notin\mathbf{I}$.

Итак, мы доказали, что не должно быть ни $\mu\cdot\langle\Delta\rangle\in\mathbf{I}$, ни $\mu\cdot\langle\Delta\rangle\notin\mathbf{I}$, чего быть не может. Следовательно, наше допущение было не верно, и $\mathtt{P}$ ***safe*** $\mathbf{I}$ ***after*** μ.

Тем самым, доказана гипотеза о безопасности: $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$.

2.2. Теперь докажем выполнение тестируемого условия: $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$.

Допустим, это не так: $\exists\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})\cap\mathbf{I}\ \exists\mathtt{P}\in\mathfrak{R}$

$\mathtt{P}$ ***safe*** $\mathbf{\Sigma}$ ***after*** μ & $\boldsymbol{obs}(\mu,\mathtt{P},\mathbf{I})\not\subseteq\boldsymbol{obs}(\mu,\mathtt{P},\mathbf{\Sigma})$.

Тогда найдётся такой символ $\mathtt{u}\in\boldsymbol{obs}(\mu,\mathtt{P},\mathbf{I})$, что $\mathtt{u}\notin\boldsymbol{obs}(\mu,\mathtt{P},\mathbf{\Sigma})$.

Тогда $\mu\cdot\langle\mathtt{u}\rangle\in\mathbf{I}$ и, следовательно, $\mu\cdot\langle\mathtt{u}\rangle\preccurlyeq_{\mathfrak{R}}\mathbf{\Sigma}$.

Рассмотрим шесть возможных случаев.

2.2.1. $\mu\cdot\langle\mathtt{u}\rangle\in\mathbf{\Sigma}$. Это противоречит $\mathtt{u}\notin\boldsymbol{obs}(\mu,\mathtt{P},\mathbf{\Sigma})$.

2.2.2. $\langle\gamma\rangle\in\mathbf{\Sigma}$. Это противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

2.2.3. $\mu\cdot\langle\mathtt{u}\rangle\preccurlyeq_{\mathfrak{R}}^{\Delta}\mathbf{\Sigma}$ = $\exists\pi\ \pi\le\mu\cdot\langle\mathtt{u}\rangle$ & $\mu\cdot\langle\mathtt{u}\rangle\ne\pi\cdot\langle\gamma\rangle$ & $\pi\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$. Поскольку $\mathtt{u}\ne\gamma$, должно быть $\pi\le\mu$ и $\pi\cdot\langle\Delta\rangle\in\mathbf{\Sigma}$. Для $\pi=\mu$ это противоречит $\mathtt{P}$ ***safe*** $\mathbf{\Sigma}$ ***after*** μ, а для $\pi<\mu$ это противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

2.2.4. $\mu\cdot\langle\mathtt{u}\rangle\preccurlyeq_{\mathfrak{R}}^{\Delta\gamma}\mathbf{\Sigma}$ = $\exists\pi\ \mu\cdot\langle\mathtt{u}\rangle=\pi\cdot\langle\Delta\rangle$ & $\forall\mathtt{R}\in\mathfrak{R}\ \exists\mathtt{t}\in\mathtt{R}\ \pi\cdot\langle\mathtt{t},\gamma\rangle\in\mathbf{\Sigma}$. Но этого не может быть, так как $\mathtt{u}\ne\Delta$.

2.2.5. $\mu\cdot\langle\mathtt{u}\rangle\preccurlyeq_{\mathfrak{R}}^{\mathtt{P}}\mathbf{\Sigma}$ = $\exists\pi\ \exists\mathtt{R}\in\mathfrak{R}\ \pi\cdot\langle\mathtt{R}\rangle\le\mu\cdot\langle\mathtt{u}\rangle$ & $\exists\mathtt{t}\in\mathtt{R}\ \pi\cdot\langle\mathtt{t},\gamma\rangle\in\mathbf{\Sigma}$. Для $\pi\cdot\langle\mathtt{R}\rangle=\mu\cdot\langle\mathtt{u}\rangle$ будет $\mathtt{R}=\mathtt{u}=\mathtt{P}$ и $\pi=\mu$, что противоречит $\mathtt{P}$ ***safe*** $\mathbf{\Sigma}$ ***after*** μ. Для $\pi\cdot\langle\mathtt{R}\rangle\le\mu$ это влечёт $\mathtt{R}$ ~~***safe***~~ $\mathbf{\Sigma}$ ***after*** π, что противоречит $\mu\in\boldsymbol{Safe}(\mathbf{\Sigma})$.

2.2.6. $\mu\cdot\langle u\rangle \preccurlyeq^{z}_{\mathfrak{R}} \mathbf{\Sigma}$ = $\exists\pi$ $\exists v\in L$ $\pi\cdot\langle v\rangle\leq\mu\cdot\langle u\rangle$ & $\forall R\in\mathfrak{R}$ ($v\in R$ $\Rightarrow$ $\exists t\in R$ $\pi\cdot\langle t,\gamma\rangle\in\mathbf{\Sigma}$). Для $\pi\cdot\langle v\rangle=\mu\cdot\langle u\rangle$ будет $v=u$, что противоречит P ***safe*** $\mathbf{\Sigma}$ ***after*** μ, а для $\pi<\mu$ это противоречит $\mu\in$***Safe***$(\mathbf{\Sigma})$.

Мы пришли к противоречию во всех шести случаях, следовательно, наше допущение было не верно, и тестируемое условие выполнено.

Тем самым, конформность доказана: $\mathbf{I}$ ***saco***$_{\mathfrak{R}}$ $\mathbf{\Sigma}$.

58. Доказательство Леммы 34

Первая часть утверждения. Из любого ϕ-маршрута, имеющего трассу без повторных ϕ-символов, можно получить нормальный ϕ-маршрут, удалением τ-циклов, τ-постфикса и повторных ϕ-петель. Очевидно, трасса исходного ϕ-маршрута получается из трассы нормального ϕ-маршрута удалением некоторых ϕ-символов. При этом, если трасса исходного ϕ-маршрута была конечной, то трасса нормального ϕ-маршрута также будет конечной.

Вторая часть утверждения. Любой ϕ-маршрут получается из некоторого ϕ-маршрута без повторных ϕ-петель повторением некоторых уже имеющихся в ϕ-маршруте ϕ-петель. При этом конечный ϕ-маршрут может быть получен из конечного ϕ-маршрута без повторных ϕ-петель.

59. Доказательство Леммы 35

1. Прежде всего, заметим, что каждое старое состояние s становится нестабильным в LTS $\mathbf{s}_{\text{norm}}$: либо оно было нестабильным в LTS $\mathbf{s}$, либо было стабильным в LTS $\mathbf{s}$, но стало нестабильным в LTS $\mathbf{s}_{\text{norm}}$ после добавления перехода $s\text{—}\tau\rightarrow s_{\text{norm}}$. Каждое новое состояние s_{norm} стабильно в LTS $\mathbf{s}_{\text{norm}}$, поскольку старое состояние s было стабильным в LTS $\mathbf{s}$.

Поэтому каждому ϕ-маршруту без повторных ϕ-петель из $\boldsymbol{R}_{\phi}^{\omega}(\mathbf{s}_{\text{norm}})$ можно однозначно сопоставить ϕ-маршрут без повторных ϕ-петель из $\boldsymbol{R}_{\phi}^{\omega}(\mathbf{s})$ следующим образом:

a) каждый переход $s \text{—} z \rightarrow s`$ сохраняется,

b) τ-переход $s \text{—} \tau \rightarrow s_{\text{norm}}$ и далее переход $s_{\text{norm}} \text{—} o \rightarrow s_{\text{norm}}$, где o ϕ-символ, заменяются на переход $s \text{—} o \rightarrow s$,

c) переход $s_{\text{norm}} \text{—} z \rightarrow s`$, где $z$ внешнее действие, заменяется на переход $s \text{—} z \rightarrow s`$.

Во всех случаях не меняется трасса ϕ-маршрута. Поэтому множество ϕ-трасс без повторных ϕ-символов не увеличивается при нормализации. Оно также не уменьшается, поскольку любой ϕ-маршрут без повторных ϕ-петель из $\boldsymbol{R}_{\phi}^{\omega}(\mathbf{s})$ указанным образом соответствует некоторому ϕ-маршруту без повторных ϕ-петель из $\boldsymbol{R}_{\phi}^{\omega}(\mathbf{s}_{\text{norm}})$.

2. В нормальной ϕ-трассе нет повторных ϕ-символов по определению. Нам нужно показать обратное: если в ϕ-трассе LTS $\mathbf{s}_{\text{norm}}$ нет повторных ϕ-символов, то она нормальна.

Пусть ϕ-трасса $\boldsymbol{\mu}$ без повторных ϕ-символов является трассой ϕ-маршрута $p \in \boldsymbol{R}_{\phi}^{\omega}(\mathbf{s}_{\text{norm}})$, который не является нормальным ϕ-маршрутом. Это значит, что нарушено требование нормальности: ϕ-маршрут содержит τ-циклы, имеет непустой τ-постфикс или проходит через некоторые стабильные состояния, но не проходит по ϕ-петлям этих состояний. Удаление τ-циклов и τ-постфикса, очевидно, не меняет трассу ϕ-маршрута.

Рассмотрим все состояния, ϕ-петли которых не включены в ϕ-маршрут. Этими состояниями могут быть только новые состояния s_{norm}, в каждое из которых ведёт единственный переход $s \text{—} \tau \rightarrow s_{\text{norm}}$. После удаления τ-

постфикса ϕ-маршрут либо бесконечный, либо заканчивается в старом состоянии, так как иначе последним переходом ϕ-маршрута являлся бы τ-переход. Поэтому, если ϕ-маршрут проходит через состояние s_{norm}, то в ϕ-маршруте имеется следующий переход $s_{norm}—z→s`$. Теперь заменим в ϕ-маршруте два перехода $s—\tau→s_{norm}—z→s`$ на один переход $s—z→s`$, который, по построению, должен быть в LTS $\mathbf{s}_{norm}$. Теперь ϕ-маршрут проходит не через новое, а через старое состояние, которое нестабильно и поэтому не имеет ϕ-символа.

Очевидно, мы получим ϕ-маршрут, имеющий ту же трассу $\boldsymbol{\mu}$, но теперь он является нормальным ϕ-маршрутом в LTS $\mathbf{s}_{norm}$, что и требовалось показать.

60. **Доказательство Леммы 36**

1. Сначала покажем, что множество нормальных ϕ-трасс канонической LTS префикс-замкнуто.

Пусть ϕ-трасса $\boldsymbol{\lambda}$ нормальна. Нам надо показать, что конечный префикс $\boldsymbol{\mu}<\boldsymbol{\lambda}$ также нормален.

Рассмотрим нормальный ϕ-маршрут $P_{\boldsymbol{\lambda}}$, имеющий трассу $\boldsymbol{\lambda}$.

Тогда у него должен быть префикс $P<P_{\boldsymbol{\lambda}}$, имеющий трассу $\boldsymbol{\mu}$.

Как префикс нормального ϕ-маршрута он не может содержать τ-циклов, повторных ϕ-петель, а также пропущенных ϕ-петель, кроме, быть может, ϕ-петли конечного состояния.

Поэтому, если ϕ-маршрут P не нормален, то возможны два случая: 1) пропущена ϕ-петля конечного состояния, 2) имеется непустой постфикс τ-переходов.

1.1. Пропущена ϕ-петля конечного стабильного состояния s.

Тогда ϕ-маршрут P продолжается в ϕ-маршруте $P_{\boldsymbol{\lambda}}$ ϕ-петлёй $s—\phi(s)→s$.

В канонической LTS начальное состояние и постсостояние каждого достижимого перехода по внешнему действию нестабильны.

Также в ϕ-трассах разрушение и дивергенция могут быть только последними символами.

Поэтому в ϕ-маршруте P предыдущим переходом должен быть τ-переход $s` \text{—} \tau \rightarrow s$.

Заметим, что в канонической LTS τ-переход, если это не τ-петля, может вести только в такое нестабильное состояние, в котором определён единственный γ-переход.

Поскольку τ-петель в ϕ-маршруте P нет, перед переходом $s` \text{—} \tau \rightarrow s$ в ϕ-маршруте P не может быть τ-перехода.

Следовательно, удаляя из ϕ-маршрута P переход $s` \text{—} \tau \rightarrow s$, получим ϕ-маршрут P_{μ} с той же трассой **μ**, но теперь уже нормальный: конечное состояние теперь нестабильно, и нет постфикса τ-переходов.

1.2. Имеется непустой постфикс τ-переходов.

Удаляя этот постфикс, очевидно, получаем ϕ-маршрут P_{μ} с той же трассой **μ**, но теперь уже нормальный: конечное состояние нестабильно (в нём начинается непустая цепочка τ-переходов), и нет постфикса τ-переходов.

2. Теперь покажем, что каноническая LTS ϕ-детерминирована.

Допустим это не так. Тогда найдутся такие два разных нормальных ϕ-маршрута P_1 и P_2, которые имеют одну трассу **μ**.

Пусть P – максимальный общий префикс этих ϕ-маршрутов, то есть эти ϕ-маршруты продолжаются после общего префикса разными переходами $s \text{—} a \rightarrow s_1$ и $s \text{—} b \rightarrow s_2$, где $\omega(P) = s$.

Символы a и b не могут быть разными символами в алфавите ϕ-трасс (ϕ-символами, внешними действиями, дивергенцией или разрушением), так как тогда трассы этих ϕ-маршрутов были бы разными.

Если это один ϕ-символ $a=b\in\Phi(L)$, то $s_1=s_2$, так как в состоянии s определена только одна ϕ-петля, и переходы $s\text{—}a\rightarrow s_1$ и $s\text{—}b\rightarrow s_2$ совпадают.

Если это дивергенция или разрушение $a=b\in\{\Delta,\gamma\}$, то также $s_1=s_2$, так как переходы по дивергенции и разрушению заканчиваются в одном состоянии $\mathbf{t}$ после преобразования $\boldsymbol{L2L}_{\phi}$.

Если это одно внешнее действие $a=b\in L$, то также $s_1=s_2$, так как в канонической LTS в состоянии s не может быть определено два разных перехода по одному внешнему действию.

Следовательно, хотя бы один из символов a или b является внутренним, для определённости, пусть $a=\tau$.

Так как в нормальном ϕ-маршруте нет τ-циклов, $s_1\neq s$.

А тогда в канонической LTS должно быть либо $s_1\text{—}\gamma\rightarrow$, либо состояние s_1 стабильно.

Пусть $s_1\text{—}\gamma\rightarrow$.

Тогда в канонической LTS в состоянии s_1 определён только γ-переход, то есть трасса ϕ-маршрута P продолжается разрушением в трассе ϕ-маршрута P_1.

Что должно быть, чтобы трасса ϕ-маршрута P также продолжалась разрушением в трассе ϕ-маршрута P_2?

Возможны два варианта: a) $b=\gamma$, b) $b=\tau$.

В случае a) в канонической LTS не может быть $s \xrightarrow{\tau} s_1 \xrightarrow{\gamma}$ и $s \xrightarrow{\gamma} s_2$.

В случае b) в канонической LTS не может быть $s_2 = s$, так как τ-петли не должно быть в нормальном ϕ-маршруте. Также состояние s_2 не может быть стабильным, так как тогда трасса ϕ-маршрута P продолжалась бы в трассе ϕ-маршрута P_2 ϕ-символом, а не разрушением. А тогда в канонической LTS остаётся единственный вариант $s \xrightarrow{\tau} s_2 \xrightarrow{\gamma}$.

Однако это также невозможно в канонической LTS, поскольку уже есть $s \xrightarrow{\tau} s_1 \xrightarrow{\gamma}$ и $s_1 \neq s_2$.

Мы пришли к противоречию, значит не может быть $s_1 \xrightarrow{\gamma}$.

Пусть состояние s_1 стабильно.

Тогда трасса ϕ-маршрута P продолжается ϕ-символом $\phi(s_1)$ в трассе нормального ϕ-маршрута P_1.

Поэтому символ b не может быть внешним действием, дивергенцией или разрушением, так как тогда трасса ϕ-маршрута P не продолжалась бы ϕ-символом $\phi(s_1)$ в трассе нормального ϕ-маршрута P_2.

С другой стороны, символ b не может быть ϕ-символом, так как в нестабильном состоянии s нет ϕ-символа.

Следовательно, $b = \tau$.

Если $s_2 \xrightarrow{\gamma}$, то в этом случае в канонической LTS в состоянии s_2 определён только γ-переход, то есть трасса ϕ-маршрута P продолжается разрушением в трассе ϕ-маршрута P_2, а не ϕ-символом $\phi(s_1)$.

Так как в нормальном ϕ-маршруте нет τ-циклов, $s_2 \neq s$.

Следовательно, в канонической LTS состояние s_2 тоже стабильно.

Но в канонической LTS $s_1 \neq s_2 \Rightarrow \phi(s_1) \neq \phi(s_2)$, то есть трасса ϕ-маршрута P продолжается разными ϕ-символами в трассах ϕ-маршрутов P_1 и P_2.

Мы пришли к противоречию. Следовательно, наше допущение не верно и каноническая LTS ϕ-детерминирована.

61. **Доказательство Леммы 37**

Пусть ϕ-трасса $\boldsymbol{\mu}$ является пределом бесконечно возрастающей цепочки ϕ-трасс $\boldsymbol{\mu}_1 < \boldsymbol{\mu}_2 < \ldots$.

Если ϕ-трасса $\boldsymbol{\mu}$ имеет бесконечный постфикс из одного ϕ-символа o, то достаточно рассмотреть префикс $\boldsymbol{\mu}_n$, завершающийся ϕ-символом o. Любой ϕ-маршрут с трассой $\boldsymbol{\mu}_n$ заканчивается в состоянии с ϕ-петлёй по ϕ-символу o. Бесконечно продолжая этот ϕ-маршрут этой ϕ-петлёй, мы получим ϕ-маршрут с ϕ-трассой $\boldsymbol{\mu}$.

В противном случае ϕ-трасса $\boldsymbol{\mu}$ имеет бесконечную подтрассу внешних действий. Удалим из ϕ-трассы $\boldsymbol{\mu}$ и всех префиксов $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots$ все повторные ϕ-символы (любую цепочку ϕ-символов заменим на один ϕ-символ), обозначая результат штрихом “`”. Очевидно ϕ-трасса $\boldsymbol{\mu}`$ будет пределом бесконечно возрастающей цепочки префиксов $\boldsymbol{\mu}`_1 < \boldsymbol{\mu}`_2 < \ldots$. Все эти ϕ-трассы не имеют повторных ϕ-символов и, поскольку LTS нормальна, все эти ϕ-трассы нормальны.

По Лемме 36, каноническая LTS ϕ-детерминирована. Следовательно, для каждой ϕ-трассы $\boldsymbol{\mu}`_i$ существует только один нормальный ϕ-маршрут $P`_i$ с трассой $\boldsymbol{\mu}`_i$. Таким образом, мы имеем бесконечно возрастающую цепочку ϕ-маршрутов $P`_1 < P`_2 < \ldots$. Тем самым, имеется бесконечный ϕ-маршрут $P`$ с трассой $\boldsymbol{\mu}`$. Добавляя обратно в нужных местах этого ϕ-маршрута ϕ-петли, получим ϕ-маршрут P с трассой $\boldsymbol{\mu}$.

62. **Доказательство Леммы 38**

Маршрутная LTS является деревом по построению: добавление ненулевого числа переходов в конец маршрута никогда не может дать исходный маршрут.

Между состояниями LTS $\boldsymbol{Run}(\mathbf{s})$ и $\mathbf{s}$ можно установить соответствие, если каждому маршруту LTS $\mathbf{s}$, поставить в соответствие его конечное состояние. Тогда для каждого перехода по действию u, определённому в состоянии одной LTS, в любом соответствующем состоянии определён такой переход по этому же действию u, что постсостояния этих переходов тоже соответствуют друг другу[36]. Отсюда следует, что соответствующие состояния либо оба нестабильны, либо оба стабильны и имеют одинаковые ϕ-символы. А тогда каждому ϕ-маршруту одной LTS соответствует ϕ-маршрут другой LTS с той же самой трассой. Тем самым, LTS $\boldsymbol{Run}(\mathbf{s})$ и $\mathbf{s}$ имеют одинаковые множества ϕ-трасс.

63. **Доказательство Леммы 39**

Рассмотрим маршрутную LTS $\boldsymbol{Run}(\mathbf{s})$.

Пусть P произвольный маршрут LTS $\boldsymbol{Run}(\mathbf{s})$.

Определим для каждого состояния $s \in V_{\boldsymbol{Run}(\mathbf{s})}$ и каждого действия $u \in L_{\gamma\tau}$ множество переходов $E(P,s,u)$ следующим образом:

- Если существует переход $(s,u,s`) \in \boldsymbol{Im}(P)$, то, поскольку LTS $\boldsymbol{Run}(\mathbf{s})$ дерево, такой переход единственный. Тогда определим множество состоящим из одного этого перехода:
  $E(P,s,u)=\{(s,u,s`)\}$.
- В противном случае множество состоит из всех переходов из s по u: $E(P,s,u)=\{(s,u,s`) \mid (s,u,s`) \in E_{\boldsymbol{Run}(\mathbf{s})}\}$.

---

[36] Такое соответствие называется строгой бисимуляцией (*strong bisimulation*), а LTS строго-бисимулянтными.

Рассмотрим семейство всех непустых множеств $E(P,s,u)$.

По аксиоме выбора, мы можем выбрать по одному элементу из каждого такого множества $E(P,s,u)$. Полученное множество переходов обозначим через $E(P)$.

Если в LTS $\boldsymbol{Run}(\mathbf{s})$ оставить только переходы из $E(P)$, то мы получим под-LTS, которую обозначим через $\mathbf{s}(P)$.

Множество всех ϕ-трасс всех таких LTS $\mathbf{s}(P)$, где P пробегает множество всех маршрутов, обозначим $\mathbf{\Sigma}=\cup\{\mathbf{\Phi}^{\omega}\circ\mathbf{s}(P)\mid P\in\boldsymbol{R}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})\}$.

При построении под-LTS $\mathbf{s}(P)$ мы удаляли в каждом состоянии s только такой переход по действию z, для которого оставался переход из s по тому же действию z: если множество $E(P,s,u)$ было не пусто, мы выбирали в нём ровно один элемент.

Следовательно, в под-LTS $\mathbf{s}(P)$ каждое состояние имеет тот же ϕ-символ, который он имел в LTS $\boldsymbol{Run}(\mathbf{s})$.

Поэтому $\forall P\in\boldsymbol{R}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})\ \ \mathbf{\Phi}^{\omega}\circ\mathbf{s}(P)\subseteq\mathbf{\Phi}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})$ и $\mathbf{\Sigma}\subseteq\mathbf{\Phi}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})$.

Также все переходы маршрута P сохранятся в под-LTS $\mathbf{s}(P)$. Следовательно, в LTS $\mathbf{s}(P)$ имеет место $\mathbf{\Phi}^{\omega}(P)\subseteq\mathbf{\Phi}^{\omega}\circ\mathbf{s}(P)$.

Поскольку это верно для каждого $P\in\boldsymbol{R}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})$, имеем $\cup\circ\mathbf{\Phi}^{\omega}\circ\boldsymbol{R}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})\subseteq\mathbf{\Sigma}$.

Поскольку $\mathbf{\Phi}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})=\cup\circ\mathbf{\Phi}^{\omega}\circ\boldsymbol{R}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})$ и, по Лемме 38, $\mathbf{\Phi}^{\omega}\circ\boldsymbol{Run}(\mathbf{s})=\mathbf{\Phi}^{\omega}(\mathbf{s})$, имеем: $\mathbf{\Sigma}=\mathbf{\Phi}^{\omega}(\mathbf{s})$.

Для завершения доказательства достаточно отметить, что каждая под-LTS $\mathbf{s}(P)$, по построению, ЛК-ветвящаяся (в каждом достижимом состоянии не более одного перехода по каждому действию).

64. Доказательство Леммы 40

Если LTS $\mathbf{s}$ ЛК-ветвящаяся, то LTS $\boldsymbol{L2L}_{\phi}(\mathbf{s})$ также ЛК-ветвящаяся: в каждом состоянии добавляется не более одного перехода по каждому ϕ-символу и по символу Δ.

Пусть ϕ-трасса μ является пределом бесконечно возрастающей цепочки ϕ-трасс $\mu_1<\mu_2<\ldots$.

Рассмотрим все конечные ϕ-маршруты, трассы которых являются префиксами ϕ-трассы μ. Очевидно, множество P этих ϕ-маршрутов является деревом.

Поскольку LTS $L2L_\phi(s)$ ЛК-ветвящаяся, дерево P К-ветвящееся.

Поскольку бесконечная ϕ-трасса имеет бесконечное множество префиксов, по Теореме Кёнига [109], в дереве P есть бесконечный маршрут, который, очевидно, в LTS $L2L_\phi(s)$ имеет в качестве трассы ϕ-трассу μ.

Следовательно, этот бесконечный маршрут является ϕ-маршрутом LTS s.

Следовательно, ϕ-трасса μ является ϕ-трассой LTS s.

65. Доказательство Теоремы 25

Множество ϕ-трасс LTS s – это множество трасс LTS $L2L_\phi(s)$. Поэтому оно является деревом.

1. Допустимость следует из того, что в LTS $L2L_\phi(s)$ переход по разрушению или дивергенции ведёт в терминальное состояние t.

2. Согласованность следует из того, что в LTS $L2L_\phi(s)$ переход по ϕ-символу o 1) является петлёй, 2) определяется в стабильном состоянии (из него не ведут τ- и γ-переходы) 3) при отсутствии переходов из этого состояния по внешним действиям из o_Γ. Из 1) следует, что после ϕ-символа не может следовать другой ϕ-символ. Из 2) следует, что после ϕ-символа не может быть дивергенции и разрушения. Из 3) следует, что после ϕ-символа не может быть внешнего действия из o_Γ.

3. Конвергентность. Если ϕ-трасса не содержит дивергенции и разрушения, она не заканчивается в состоянии t, то есть заканчивается в некотором множестве состояний LTS s. Если ϕ-трасса заканчивается только в нестабильных состояниях, то из каждого такого состояния выходит γ- или τ-

переход. Следовательно, хотя бы в одном состоянии должен быть γ-переход или начинаться бесконечный маршрут τ-переходов (состояние дивергентно). В этом случае ϕ-трасса продолжается разрушением и/или дивергенцией. Поэтому, если ϕ-трасса не содержит и не продолжается дивергенцией и разрушением, она заканчивается хотя бы в одном стабильном состоянии. А тогда ϕ-трасса продолжается ϕ-символом этого стабильного состояния.

4. Замкнутость по ***dr***-операции следует из того, что в LTS $\boldsymbol{L2L_{\phi}(\mathbf{s})}$ переход по ϕ-символу o является петлёй.

5. Полнота (замкнутость по ***a***-операции). Если ϕ-трасса имеет вид $\boldsymbol{\mu}\cdot\langle o\rangle$, где o ϕ-символ, то она заканчивается в таком стабильном состоянии s, что $\phi(s)=o$. Если $z\in L\setminus o_r$, то из состояния s должен быть переход по действию z, а тогда есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o, z\rangle$. Если $z\in o_g$, то из состояния s должен быть переход по действию z, ведущий в постсостояние с γ-переходом, а тогда есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o, z, \gamma\rangle$.

6. Предельность. По Лемме 39, существует множество ЛК-ветвящихся LTS-моделей, объединение множеств ϕ-трасс которых равно множеству ϕ-трасс LTS $\mathbf{s}$. По Лемме 40, множество ϕ-трасс каждого LTS-компонента предельно, а, по доказанному выше, удовлетворяет первым пяти свойствам ϕ-модели. Следовательно, множество ϕ-трасс каждого LTS-компонента является предельной ϕ-моделью. Следовательно, множество ϕ-трасс LTS $\mathbf{s}$ равно объединению предельных ϕ-моделей, то есть является ϕ-моделью.

66. **Доказательство Леммы 41**

1. Пусть состояние LTS $\boldsymbol{\Phi 2L(\Sigma)}$ имеет вид $\boldsymbol{\mu}$, где $\boldsymbol{postf(\mu)}=\epsilon$. Тогда ϕ-трасса $\boldsymbol{\mu}$ не содержит дивергенции и разрушения. Следовательно, по конвергентности ϕ-модели, ϕ-трасса $\boldsymbol{\mu}$ продолжается в $\boldsymbol{\Sigma}$ дивергенцией,

разрушением или некоторым ϕ-символом o. А тогда, по правилам вывода, в состоянии $\boldsymbol{\mu}$ определена τ- или γ-переход. В любом случае состояние $\boldsymbol{\mu}$ нестабильно.

2. Пусть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o\rangle$ заканчивается ϕ-символом o. Тогда, по согласованности ϕ-модели, она не продолжается в $\boldsymbol{\Sigma}$ дивергенцией или разрушением. Поскольку все состояния LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ это ϕ-трассы без повторных ϕ-символов, для любого ϕ-символа $o`$ ϕ-трасса $\boldsymbol{\mu}\cdot\langle o, o`\rangle$ не является состоянием LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$. Следовательно, по правилам вывода, из состояния $\boldsymbol{\mu}\cdot\langle o\rangle$ не выходят τ- и γ-переходы, то есть оно стабильно.

Пусть $z\in L$. Если $\boldsymbol{\mu}\cdot\langle o, z\rangle\in\boldsymbol{\Sigma}$, то ϕ-трасса $\boldsymbol{\mu}\cdot\langle o, z\rangle$ не имеет повторных ϕ-символов. Поэтому, согласно правилам вывода, переход $\boldsymbol{\mu}\cdot\langle o\rangle\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle o, z\rangle$ имеется тогда и только тогда, когда $\boldsymbol{\mu}\cdot\langle o, z\rangle\in\boldsymbol{\Sigma}$. По согласованности и полноте ϕ-модели, $\boldsymbol{\mu}\cdot\langle o, z\rangle\in\boldsymbol{\Sigma}$ тогда и только тогда, когда $z\notin o_r$. Следовательно, $\phi(\boldsymbol{\mu}\cdot\langle o\rangle)_r=o_r$.

Далее, согласно правилам вывода, переход $\boldsymbol{\mu}\cdot\langle o, z\rangle\text{—}\gamma\rightarrow\boldsymbol{\mu}\cdot\langle o, z\rangle$ имеется тогда и только тогда, когда в $\boldsymbol{\Sigma}$ есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o, z\rangle$ и $z\in o_g$. Следовательно, $\phi(\boldsymbol{\mu}\cdot\langle o\rangle)_g=o_g$.

Тем самым, $\phi(\boldsymbol{\mu}\cdot\langle o\rangle)=o$, что и требовалось доказать.

67. **Доказательство Леммы 42**

Все свойства канонической LTS непосредственно следуют из правил вывода и Леммы 41.

68. **Доказательство Леммы 43**

0. Сначала докажем, что все состояния $\boldsymbol{\mu}$ достижимы. Допустим, некоторое состояние $\boldsymbol{\mu}$ недостижимо. Выберем ϕ-трассу $\boldsymbol{\mu}$ минимальной длины. Поскольку начальное состояние ϵ достижимо, ϕ-трассу $\boldsymbol{\mu}$ можно представить в виде $\boldsymbol{\mu}=\boldsymbol{\mu}`\cdot\langle u\rangle$, где состояние $\boldsymbol{\mu}`$ достижимо. Поскольку $\boldsymbol{\mu}$ не завершается символом Δ или γ, $u=z\in L$ или $u=o\in\Phi(L)$. Но тогда, по правилам вывода, есть переход $\boldsymbol{\mu}`\text{—}z\rightarrow\boldsymbol{\mu}`\cdot\langle z\rangle$ или $\boldsymbol{\mu}`\text{—}\tau\rightarrow\boldsymbol{\mu}`\cdot\langle o\rangle$. Следовательно состояние $\boldsymbol{\mu}=\boldsymbol{\mu}`\cdot\langle u\rangle$ достижимо, что противоречит допущению.

1. Теперь докажем, что в каждом достижимом состоянии $\boldsymbol{\mu}$ заканчивается нормальная ϕ-трасса $\boldsymbol{\mu}$.

В начальном состоянии ϵ, очевидно, заканчивается пустая ϕ-трасса, являющаяся трассой пустого ϕ-маршрута.

Поскольку, по Лемме 41, начальное состояние нестабильно, пустой ϕ-маршрут нормален и, следовательно, пустая ϕ-трасса нормальна.

Далее по индукции: пусть в состоянии $\boldsymbol{\mu}$ заканчивается нормальный ϕ-маршрут P, трассой которого является ϕ-трасса $\boldsymbol{\mu}$.

Рассмотрим переходы из этого состояния в другие состояния, определяемые правилами вывода.

1.1. Правило вывода (1). Пусть есть переход $\boldsymbol{\mu}\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle z\rangle$, где $z\in L$.

Поскольку, по Лемме 41, состояние $\boldsymbol{\mu}\cdot\langle z\rangle$ нестабильно, ϕ-трасса $\boldsymbol{\mu}\cdot\langle z\rangle$ является трассой нормального ϕ-маршрута $P\cdot\langle\boldsymbol{\mu}\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle z\rangle\rangle$, который заканчивается в состоянии $\boldsymbol{\mu}\cdot\langle z\rangle$.

1.2. Правило вывода (2). Пусть есть переход $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle o\rangle$, где $o\in\Phi(L)$.

Поскольку, по Лемме 41, состояние $\boldsymbol{\mu}\cdot\langle\circ\rangle$ стабильно и $\phi(\boldsymbol{\mu}\cdot\langle\circ\rangle)=\circ$, ϕ-трасса $\boldsymbol{\mu}\cdot\langle\circ\rangle$ является трассой нормального ϕ-маршрута $P\cdot\langle\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle\circ\rangle\text{—}\circ\rightarrow\boldsymbol{\mu}\cdot\langle\circ\rangle\rangle$, который заканчивается в состоянии $\boldsymbol{\mu}\cdot\langle\circ\rangle$.

1.3. Правила вывода (3) и (4) определяют переходы в то же самое состояние, поэтому этот случай можно не рассматривать.

1.4. Правило вывода (5) определяет переход в состояние $\mathbf{t}$ (а не $\boldsymbol{\mu}$), поэтому этот случай можно не рассматривать.

2. Теперь докажем, что каждый конечный нормальный ϕ-маршрут с трассой $\boldsymbol{\mu}$, не завершающейся символом Δ или γ, заканчивается в единственном состоянии $\boldsymbol{\mu}$.

По Лемме 42, LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ каноническая. Тогда, по Лемме 36, она ϕ-детерминирована: каждая нормальная ϕ-трасса $\boldsymbol{\mu}$ является трассой единственного нормального ϕ-маршрута. Нам надо показать, что этот ϕ-маршрут заканчивается в состоянии $\boldsymbol{\mu}$, если конечная ϕ-трасса $\boldsymbol{\mu}$ не завершается символом Δ или γ.

Начальное состояние ϵ, по Лемме 41, нестабильно. Следовательно, пустой ϕ-маршрут, начинающийся и заканчивающийся в начальном состоянии, нормален и имеет трассу ϵ.

Далее по индукции: пусть нормальный ϕ-маршрут P с трассой $\boldsymbol{\mu}$ заканчивается в единственном состоянии $\boldsymbol{\mu}$. Рассмотрим ближайшие продолжения этой ϕ-трассы, оставляющие её нормальной и отличные от символов Δ и γ.

2.1. Продолжение внешним действием символом $z\in L$: $\boldsymbol{\mu}=\langle z\rangle\Rightarrow\boldsymbol{\lambda}$.

По правилам вывода, любой τ-переход, отличный от τ-петли (не входящей в нормальный ϕ-маршрут), ведёт либо а) в состояние, в котором определён

только γ-переход, либо b) в состояние, которое, по Лемме 41, стабильно, то есть имеет ϕ-символ.

В случае a) трасса ϕ-маршрута продолжается только разрушением, а в случае b) только ϕ-символом.

Поэтому должно быть $\boldsymbol{\mu}\text{—}z\rightarrow\boldsymbol{\lambda}$.

По правилу вывода (1), $\boldsymbol{\lambda}=\boldsymbol{\mu}\cdot\langle z\rangle$.

По Лемме 41, состояние $\boldsymbol{\mu}\cdot\langle z\rangle$ нестабильно.

Следовательно, ϕ-маршрут $P`=P\cdot\langle\boldsymbol{\mu}\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle z\rangle\rangle$ нормален и имеет трассу $\boldsymbol{\mu}\cdot\langle z\rangle$.

2.2. Продолжение ϕ-символом $o\in\Phi(L)$: $\boldsymbol{\mu}\Longrightarrow\boldsymbol{\lambda}$ & $\phi(\boldsymbol{\lambda})=o$.

В нормальной ϕ-трассе не может быть двух ϕ-символов подряд, поэтому состояние $\boldsymbol{\mu}$ нестабильно, а состояние $\boldsymbol{\lambda}$ стабильно (так как имеет ϕ-символ), и, следовательно, $\boldsymbol{\mu}\neq\boldsymbol{\lambda}$.

По правилам вывода, любой τ-переход, отличный от τ-петли (не входящей в нормальный ϕ-маршрут), ведёт либо a) в состояние, в котором определён только γ-переход, либо b) в состояние, которое, по Лемме 41, стабильно, то есть имеет ϕ-символ.

Поэтому должно быть $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\lambda}$.

Применимо только правило вывода (2), следовательно, $\boldsymbol{\lambda}=\boldsymbol{\mu}\cdot\langle o`\rangle$, где $o`\in\Phi(L)$, а, по Лемме 41, $\phi(\boldsymbol{\mu}\cdot\langle o`\rangle)=o`$. Поэтому $o`=o$.

Следовательно, ϕ-маршрут $P`=P\cdot\langle\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle o\rangle\text{—}o\rightarrow\boldsymbol{\mu}\cdot\langle o\rangle\rangle$ нормален и имеет трассу $\boldsymbol{\mu}\cdot\langle o\rangle$.

3. Теперь докажем, что имеется конечная нормальная ϕ-трасса $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ или $\boldsymbol{\mu}\cdot\langle\gamma\rangle$ тогда и только тогда, когда в $\boldsymbol{\Sigma}$ имеется ϕ-трасса без повторных ϕ-символов $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ или $\boldsymbol{\mu}\cdot\langle\gamma\rangle$, соответственно.

Это непосредственно следует из доказанных утверждений 1 и 2, а также правил вывода (3) , (4) и (5).

69. **Доказательство Леммы 44**

По Лемме 43, множество конечных нормальных ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ совпадает с подмножеством конечных ϕ-трасс без повторных ϕ-символов ϕ-модели $\boldsymbol{\Sigma}$.

По Лемме 34, множество всех конечных ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ получается из множество её конечных нормальных ϕ-трасс с помощью ***dr***-замыкания.

По замкнутости ϕ-модели, множество конечных ϕ-трасс ϕ-модели $\boldsymbol{\Sigma}$ также получается из её множества ϕ-трасс без повторных ϕ-символов с помощью ***dr***-замыкания.

Отсюда непосредственно следует утверждение Леммы.

70. **Доказательство Леммы 45**

Поскольку нормальная ϕ-трасса не имеет повторных ϕ-символов, нам достаточно доказать обратное: любая ϕ-трасса без повторных ϕ-символов нормальна, то есть является трассой нормального ϕ-маршрута.

По Лемме 43, множество конечных нормальных ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ совпадает с множеством конечных ϕ-трасс без повторных ϕ-символов ϕ-модели $\boldsymbol{\Sigma}$. По Лемме 44, множества всех конечных ϕ-трасс этой ϕ-модели $\boldsymbol{\Sigma}$ и этой LTS также совпадают. Отсюда следует, что множество конечных нормальных ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ совпадает с множеством её конечных ϕ-трасс без

повторных ϕ-символов, то есть доказываемое утверждение верно для конечных ϕ-трасс.

Пусть ϕ-трасса $\boldsymbol{\sigma}$ бесконечна и не имеет повторных ϕ-символов.

По Лемме 42, LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ каноническая. А тогда, по Лемме 36, она ϕ-детерминированная: каждая её конечная нормальная ϕ-трасса является трассой единственного нормального ϕ-маршрута.

Рассмотрим для каждого конечного префикса $\boldsymbol{\mu}<\boldsymbol{\sigma}$ такой нормальный ϕ-маршрут $\mathtt{P}_{\boldsymbol{\mu}}$ с трассой $\boldsymbol{\mu}$. Каждый префикс $\boldsymbol{\mu}$ трассы $\boldsymbol{\sigma}$ нормального ϕ-маршрута является трассой некоторого нормального префикса ϕ-маршрута: для этого достаточно взять любой префикс ϕ-маршрута с трассой $\boldsymbol{\mu}$ и удалить его τ-постфикс. Поэтому множество маршрутов $\mathtt{P}_{\boldsymbol{\mu}}$, где $\boldsymbol{\mu}$ пробегает все префиксы бесконечной ϕ-трассы $\boldsymbol{\sigma}$, линейно упорядочено. Отсюда следует, что существует ϕ-маршрут $\mathtt{P}$, префиксом которого является каждый нормальный ϕ-маршрут $\mathtt{P}_{\boldsymbol{\mu}}$. Отсюда следует, что ϕ-маршрут $\mathtt{P}$ нормален и имеет ϕ-трассу $\boldsymbol{\mu}$.

71. **Доказательство Леммы 46**

По Лемме 42, LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ каноническая.

По Лемме 45, LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ нормальная.

Тогда, по Лемме 37, LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ предельная.

72. **Доказательство Леммы 47**

По Лемме 44 множества конечных ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ и ϕ-модели $\boldsymbol{\Sigma}$ совпадают.

По Лемме 46, LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ предельная.

Поскольку ϕ-модель $\boldsymbol{\Sigma}$ тоже предельная, множества всех ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ и ϕ-модели $\boldsymbol{\Sigma}$ являются замыканиями по пределу одного и того же множества конечных ϕ-трасс.

Следовательно, множества всех ϕ-трасс LTS $\boldsymbol{\Phi 2L}(\boldsymbol{\Sigma})$ и ϕ-модели $\boldsymbol{\Sigma}$ совпадают, что и требовалось доказать.

73. Доказательство Леммы 48

Утверждение непосредственно следует из определения объединения LTS.

74. Доказательство Теоремы 26

По определению, любая ϕ-модель $\boldsymbol{\Sigma}$ представима в виде объединения множества предельных ϕ-моделей. По Лемме 47, для каждой предельной ϕ-модели $\boldsymbol{\Sigma}_i$ можно найти LTS $\mathbf{s}_i$ с множеством ϕ-трасс $\boldsymbol{\Sigma}_i$. По Лемме 48, объединение этого множества LTS является LTS с множеством ϕ-трасс $\boldsymbol{\Sigma}$.

75. Доказательство Теоремы 27

Поскольку для каждого стабильного состояния s его ***ref***-множество $\phi(s)_r = L\backslash \boldsymbol{init}(s)$, отказы, порождаемые этим состоянием (Определение 40), и отказы, вложенные в ***ref***-множество этого ϕ-символа, совпадают:

$\mathcal{P}(\phi(s)_r) = \mathcal{P}(L\backslash \boldsymbol{init}(s)) = \boldsymbol{f}(s)$.

Отказы и ϕ-символы определены только в стабильном состоянии и моделируются петлями в этом состоянии. Поэтому ***F***-трасса LTS является ***F***-трассой некоторого конечного маршрута этой LTS, то всегда найдётся такая конечная ϕ-трасса этого маршрута, которая порождает эту ***F***-трассу. Наоборот, ***F***-трасса, порождаемая (конечной) ϕ-трассой любого конечного маршрута LTS, является одной из ***F***-трасс этого маршрута. Поскольку бесконечные ϕ-трассы не генерируют ***F***-трасс, утверждение доказано.

76. **Доказательство Леммы 49**

Обозначим: $a=\phi(k)$, $b=\phi(l)$.

Заметим, что ***init***$(k)=A\backslash a_r$ и ***init***$(l)=B\backslash b_r$.

Также заметим, что в стабильных состояниях k и l не может быть τ- и γ-переходов.

Композиционный алфавит определяется как $A\downharpoonleft\upharpoonright B = (A\backslash\underline{B})\cup(B\backslash\underline{A})$.

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: ***init***$(k)\cap$***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 $c=a\downharpoonleft\upharpoonright b$.

   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:

$c=(c_r,c_g)=((A\setminus\underline{B})\cap a_r\cup(B\setminus\underline{A})\cap b_r,(A\setminus\underline{B})\cap a_g\cup(B\setminus\underline{A})\cap b_g)=a\barwedge b$.

77. Доказательство Леммы 50

Непосредственно следует из очевидной коммутативности композиции ϕ-символов и определения композиции ϕ-трасс: симметричность операндов в правилах $(\barwedge_\phi 0,3\div6)$ и взаимная симметричность правил $(\barwedge_\phi 1)$ и $(\barwedge_\phi 2)$.

78. Доказательство Леммы 51

Пусть $A,B\subseteq Z^{\#}$, $\boldsymbol{\kappa}\in\boldsymbol{LTS}(A_\gamma)$, $\boldsymbol{\iota}\in\boldsymbol{LTS}(B_\gamma)$.

Поскольку ϕ-трасса LTS не продолжается после разрушения, нам достаточно рассматривать только такие маршруты, которые не продолжаются после перехода по разрушению.

1. $\forall K\in\boldsymbol{R}^\omega(\boldsymbol{\kappa})\ \forall L\in\boldsymbol{R}^\omega(\boldsymbol{\iota})\ \forall M\in K\barwedge L\ \forall\boldsymbol{\mu}\in\boldsymbol{\Phi}^\omega(M)$

$\exists\boldsymbol{\kappa}\in\boldsymbol{\Phi}^\omega(K)\ \exists\boldsymbol{\lambda}\in\boldsymbol{\Phi}^\omega(L)\ \boldsymbol{\mu}\in\boldsymbol{\kappa}\barwedge\boldsymbol{\lambda}$.

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 $k_0l_0,k_1l_1,\ldots$ 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 $l_0,l_1,\ldots$. Из конечности композиционного маршрута M следует конечность маршрутов-операндов K и L, а ϕ-трасса конечного маршрута

не содержит Δ-символа, поскольку Δ-символ, заканчивающий ϕ-трассу маршрута, заменяет собой бесконечный τ-постфикс маршрута. Поэтому любые ϕ-трассы **κ** (**λ**) маршрута K (L) имеют одну и ту же подпоследовательность базовых символов $\boldsymbol{\kappa}^{\downarrow}A_{\gamma}$ ($\boldsymbol{\lambda}^{\downarrow}B_{\gamma}$), следовательно, они различаются только ϕ-символами.

Выберем ϕ-трассы **κ** и **λ** маршрутов K и L так, чтобы для каждого i в ϕ-трассы **κ** и **λ** помещалось столько же ϕ-символов состояний k_i и l_i, соответственно, сколько ϕ-символов состояния $k_i l_i$ помещается в ϕ-трассу **μ**. Это всегда можно сделать, поскольку ϕ-символ состояния понимается как переход-петля в этом состоянии.

Выбор начального состояния $k_0 l_0$ соответствует композиции пустых ϕ-трасс по правилу вывода $(\wr_{\phi}\lfloor 0)$. Применение каждого из правил вывода асинхронного перехода $(\wr\lfloor 1\div 2)$ к маршрутам-операндам для символа $z\neq\tau$ добавляет в ϕ-трассу **μ** символ z, что соответствует применению соответствующего правила вывода $(\wr_{\phi}\lfloor 1\div 2)$ к ϕ-трассам-операндам. Для символа $z=\tau$ применение правил вывода $(\wr\lfloor 1\div 2)$ не меняет ϕ-трассу **μ** и ϕ-трассы-операнды. Применение правила вывода синхронного перехода $(\wr\lfloor 3)$ к маршрутам также не меняет ϕ-трассу **μ**, что соответствует применению соответствующего правила вывода $(\wr_{\phi}\lfloor 3)$ к ϕ-трассам-операндам. Учитывая описанные выше правила выбора ϕ-символов в ϕ-трассах **κ** и **λ**, размещение в ϕ-трассе **μ** ϕ-символа состояния $k_i l_i$ соответствует применению правила вывода $(\wr_{\phi}\lfloor 4)$ к ϕ-трассам-операндам, при условии, что $\phi(k_i l_i)=\phi(k_i)\wr\phi(l_i)$. Выполнение последнего условия доказано в Лемме 49.

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 $M_1 \in K_1 \wr\!\rfloor L_1$ конечных префиксов маршрутов-операндов $K=K_1 \cdot K_2$ и $L=L_1 \cdot L_2$. А тогда $M_2 \in K_2 \wr\!\rfloor L_2$.

Далее возможны два случая.

1.2.1. ϕ-трасса $\boldsymbol{\mu}$ не завершается дивергенцией (Δ-символом).

Тогда $\boldsymbol{\mu}$ является также одной из ϕ-трасс конечного префикса M_1. Поэтому, по 1.1, $\exists \boldsymbol{\kappa}_1 \in \boldsymbol{\Phi}(K_1)\ \exists \boldsymbol{\lambda}_1 \in \boldsymbol{\Phi}(L_1)\ \boldsymbol{\mu} \in \boldsymbol{\kappa}_1 \wr\!\rfloor \boldsymbol{\lambda}_1$.

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 $\boldsymbol{\kappa}_1 \in \boldsymbol{\Phi}(K)$ и $\boldsymbol{\lambda}_1 \in \boldsymbol{\Phi}(L)$. 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 $\boldsymbol{T}(K_2)=\underline{\boldsymbol{T}(L_2)} \in (A \cap \underline{B})^{\infty}$, что даёт бесконечную цепочку синхронных переходов (быть может, перемежающуюся τ-переходами в маршрутах-операндах). Тогда $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1 \cdot \boldsymbol{T}(K_2) \in \boldsymbol{\Phi}(K)$ и $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1 \cdot \boldsymbol{T}(L_2) \in \boldsymbol{\Phi}(L)$.

По правилам вывода $(\wr_\phi\!\rfloor 3)$ и $(\wr_\phi\!\rfloor 6)$, $\boldsymbol{T}(K_2) \wr\!\rfloor \boldsymbol{T}(L_2)=\{\epsilon\}$.

А тогда, по правилу индуктивного предела $(\Downarrow_\phi 6)$, $\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}$.

1.2.2. ϕ-трасса $\boldsymbol{\mu}$ завершается дивергенцией (Δ-символом).

Тогда ϕ-трассу можно представить в виде $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\Delta\rangle$, где $\boldsymbol{\mu}_1$ является одной из ϕ-трасс маршрута M_1. Поэтому, по 1.1, $\exists\boldsymbol{\kappa}_1\in\boldsymbol{\Phi}(K_1)$ $\exists\boldsymbol{\lambda}_1\in\boldsymbol{\Phi}(L_1)$ $\boldsymbol{\mu}_1\in\boldsymbol{\kappa}_1\uparrow\downarrow\boldsymbol{\lambda}_1$.

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 $\boldsymbol{\kappa}_1\cdot\langle\Delta\rangle\in\boldsymbol{\Phi}(K)$ и $\boldsymbol{\lambda}_1\in\boldsymbol{\Phi}(L)$. По правилу вывода $(\Downarrow_\phi 1)$, $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\Delta\rangle\in\boldsymbol{\kappa}_1\cdot\langle\Delta\rangle\uparrow\downarrow\boldsymbol{\lambda}_1$. Аналогично доказывается случай бесконечного постфикса L_2: по правилу $(\Downarrow_\phi 2)$, $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\Delta\rangle\in\boldsymbol{\kappa}_1\uparrow\downarrow\boldsymbol{\lambda}_1\cdot\langle\Delta\rangle$.

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 $\boldsymbol{T}(K_2)=\underline{\boldsymbol{T}(L_2)}\in(A\cap\underline{B})^\infty$, что даёт бесконечную цепочку синхронных переходов (быть может, перемежающуюся τ-переходами в маршрутах-операндах). Тогда $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1\cdot\boldsymbol{T}(K_2)\in\boldsymbol{\Phi}(K)$ и $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\boldsymbol{T}(L_2)\in\boldsymbol{\Phi}(L)$. По правилу моделирования дивергенции Δ-символом $(\Downarrow_\phi 5)$, $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\Delta\rangle\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}$.

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 $(\Phi\downarrow 6)$, имеем $\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}$.

Утверждение 1.3 доказано.

Утверждение 1 доказано.

2. $\forall K\in\boldsymbol{R}^{\omega}(\boldsymbol{\kappa})\ \ \forall L\in\boldsymbol{R}^{\omega}(\boldsymbol{\iota})\ \ \forall\boldsymbol{\kappa}\in\boldsymbol{\Phi}^{\omega}(K)\ \ \forall\boldsymbol{\lambda}\in\boldsymbol{\Phi}^{\omega}(L)$

$\forall\boldsymbol{\mu}\in\boldsymbol{\kappa}\uparrow\downarrow\boldsymbol{\lambda}\ \ \exists M\in K\uparrow\downarrow L\ \ \boldsymbol{\mu}\in\boldsymbol{\Phi}^{\omega}(M)$.

Нам достаточно рассмотреть три случая в зависимости от того, применяются или нет для построения ϕ-трассы $\boldsymbol{\mu}$ правило моделирования дивергенции Δ-символом $(\Phi\downarrow 5)$ и правило индуктивного предела $(\Phi\downarrow 6)$. Заметим, что эти два правила не могут применяться вместе для построения одной и той же ϕ-трассы $\boldsymbol{\mu}$.

2.1. Пусть правила вывода $(\Phi\downarrow 5)$ и $(\Phi\downarrow 6)$ не применяются.

ϕ-трасса $\boldsymbol{\mu}$ однозначно определяется правилом вывода $(\Phi\downarrow 0)$ и конечной последовательностью применения правил вывода $(\Phi\downarrow 1\div 4)$ к заданным ϕ-трассам-операндам $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$.

Каждому применению правила вывода $(\Phi\downarrow 1)$ для символа z к ϕ-трассам-операндам соответствует применение правила вывода $(\uparrow\downarrow 1)$ 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 $(\uparrow\downarrow 1)$ для τ и левого маршрута. Аналогично для правил вывода $(\Phi\downarrow 2)$ и $(\uparrow\downarrow 2)$.

Каждому применению правила вывода $(\Phi\downarrow 3)$ к ϕ-трассам-операндам соответствует применение правила вывода $(\uparrow\downarrow 3)$ к маршрутам-операндам. Это может не получиться, если один из маршрутов-операндов

продолжается τ-переходом. Тогда мы сначала применяем правила вывода $(\|\downarrow 1\div 2)$ 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 $M\in K\|\downarrow L\subseteq \boldsymbol{R}(\boldsymbol{\kappa}\|\downarrow\boldsymbol{\iota})$. Далее рассмотрим применение правила вывода $(\|_\phi\downarrow 4)$. Оно компонует два ϕ-символа a и b в один ϕ-символ $a\|b$, если $a\|b\neq\tau$, или не меняет компонуемую ϕ-трассу, если $a\|b=\tau$. 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: $a\|b=\tau$ влечёт нестабильность состояния kl, а, если $a\|b\neq\tau$, то состояние kl стабильно, и $\phi(kl)=a\|b$. Выполнение этого условия доказано в Лемме 49.

Утверждение 2.1 доказано.

2.2. Пусть применяется моделирование синхронной дивергенции Δ-символом $(\|_\phi\downarrow 5)$.

Это правило формирует конечную ϕ-трассу, заканчивающуюся Δ-символом. Для такой ϕ-трассы никакое правило вывода далее не может применяться.

Итак, ϕ-трассы можно представить в виде $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\Delta\rangle$, $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1\cdot\boldsymbol{\kappa}_2$, $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\boldsymbol{\lambda}_2$, где $\boldsymbol{\kappa}_2=\underline{\boldsymbol{\lambda}}_2\in(A\cap\underline{B})^\infty$ и $\boldsymbol{\mu}_1\in\boldsymbol{\kappa}_1\|\boldsymbol{\lambda}_1$. Очевидно, ϕ-трасса $\boldsymbol{\mu}_1$ получена с помощью правил вывода $(\|_\phi\downarrow 1\div 4)$. Тогда маршруты можно представить в виде $K=K_1\cdot K_2$ и $L=L_1\cdot L_2$, причём $\boldsymbol{\kappa}_1\in\boldsymbol{\Phi}(K_1)$ и $\boldsymbol{\lambda}_1\in\boldsymbol{\Phi}(L_1)$. Построим маршруты $M\in K\|\downarrow L$ и $M_1\in K_1\|\downarrow L_1$ по описанным выше однозначным

правилам, определяемым последовательностью применения правил вывода $(\Vdash_\phi\downarrow 1\div 3)$ и описанной дисциплиной выбора τ-переходов. Очевидно, $M=M_1\cdot M_2$, где M_2 – бесконечный τ-маршрут. По доказанному случаю, $\boldsymbol{\mu}_1\in\boldsymbol{\Phi}(M_1)$. Тогда $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\Delta\rangle\in\boldsymbol{\Phi}^\omega(M)$.

Утверждение 2.2 доказано.

2.3. Пусть не применяется правило вывода $(\Vdash_\phi\downarrow 5)$, но применяется правило индуктивного предела $(\Vdash_\phi\downarrow 6)$.

Этот случай отличается от случая 2.1 только тем, что, по правилу индуктивного предела, ϕ-трасса $\boldsymbol{\mu}$ формируется бесконечной последовательностью применения правил вывода $(\Vdash_\phi\downarrow 0\div 4)$. 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, $M\in K\Vdash L$ и $\boldsymbol{\mu}\in\boldsymbol{\Phi}^\omega(M)$.

Если хотя бы один из маршрутов K или L бесконечный, то маршрут M бесконечный и получается с использование правила индуктивного предела. Также $M\in K\Vdash L$ и $\boldsymbol{\mu}\in\boldsymbol{\Phi}^\omega(M)$.

Утверждение 2.3 доказано. Утверждение 2 доказано.

79. Доказательство Теоремы 28

ϕ-трассы LTS – это ϕ-трассы её маршрутов. Маршруты композиции LTS совпадают с композициями маршрутов LTS-операндов. Поэтому требуемое утверждение непосредственно следует из Леммы 51.

80. **Доказательство Теоремы 29**

Если ϕ-мажорирование «один ко многим» определено как принадлежность, то ϕ-мажорирование «многие ко многим» означает вложенность.

1. Генеративность ϕ-мажорирования: подмножество ϕ-трасс порождает подмножество множества ***F***-трасс, порождаемых ϕ-трассами надмножества.

2. Композиционность ϕ-мажорирования: объединение композиции подмножеств ϕ-трасс вложено в объединение композиции надмножеств ϕ-трасс, поскольку композиция множеств ϕ-трасс определяется как множество композиций всех попарных ϕ-трасс из этих множеств.

3. Рефлексивность ϕ-мажорирования: множество ϕ-трасс вложено само в себя.

81. **Доказательство Леммы 52**

1. Конформность преобразования.

Рассмотрим произвольную ***F***-трассу $\mu \in \boldsymbol{F} \circ \boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S})$.

По генеративности ϕ-трасс (Теорема 27), найдётся такая ϕ-трасса $\boldsymbol{\mu} \in \boldsymbol{\Phi}^{\omega} \circ \boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S})$, что $\mu \in \xi(\boldsymbol{\mu})$.

По определению преобразования, найдётся такая конформная реализация $\mathbf{I} \in \mathfrak{I}_{\mathfrak{R}}(\mathbf{S})$, что $\boldsymbol{\mu} \in \boldsymbol{\Phi}^{\omega}(\mathbf{I})$.

Следовательно, по генеративности ϕ-трасс (Теорема 27), $\mu \in \boldsymbol{F}(\mathbf{I})$.

Поскольку конформность влечёт $\mathfrak{R}$-мажорирование (Теорема 24), $\mu \preccurlyeq_{\mathfrak{R}} \boldsymbol{F}(\mathbf{S})$.

Поскольку мы выбрали ***F***-трассу μ произвольно, имеет место $\boldsymbol{F} \circ \boldsymbol{Conf}_{\mathfrak{R}}(\mathbf{S}) \preccurlyeq_{\mathfrak{R}} \boldsymbol{F}(\mathbf{S})$.

Поскольку $\mathfrak{R}$-мажорирование влечёт конформность (Теорема 24), $\boldsymbol{Conf}_{\mathfrak{R}}(\mathtt{S})$ $\boldsymbol{saco}_{\mathfrak{R}}$ $\mathtt{S}$.

2. Мажорантность преобразования

По определению преобразования, для каждой конформной реализации $\mathtt{I}\in\mathfrak{I}_{\mathfrak{R}}(\mathtt{S})$ имеет место $\boldsymbol{\Phi}^{\omega}(\mathtt{I})\subseteq\boldsymbol{\Phi}^{\omega}\circ\boldsymbol{Conf}_{\mathfrak{R}}(\mathtt{S})$.

Тогда, по определению ϕ-мажорирования (Определение 109), $\boldsymbol{\Phi}^{\omega}(\mathtt{I})\preccurlyeq\boldsymbol{\Phi}^{\omega}\circ\boldsymbol{Conf}_{\mathfrak{R}}(\mathtt{S})$, что и требовалось доказать.

82. **Доказательство Леммы 53**

Объединение дееревьев ϕ-трасс, очевидно, является деревом ϕ-трасс.

Допустимость и согласованность ϕ-трассы не зависит от множества, которому она принадлежит. Поскольку компоненты объединения допустимы и согласованы, объединение также допустимо и согласовано.

Конвергентность. Если ϕ-трасса объединения не содержит и не продолжается разрушением и дивергенцией, то это верно и в том компоненте, которому эта ϕ-трасса принадлежит. Следовательно, в этом компоненте ϕ-трасса продолжается ϕ-символом. А тогда она продолжается этим ϕ-символом и в объединении.

Замкнутость. Каждая ϕ-трасса объединения принадлежит некоторому компоненту. Следовательно, в этот компонент вложено ***dr***-замыкание ϕ-трассы. А тогда это ***dr***-замыкание вложено и в объединение.

Полнота. Каждая ϕ-трасса объединения принадлежит некоторому компоненту. Следовательно, в этот компонент вложено ***a***-замыкание ϕ-трассы. А тогда это ***a***-замыкание вложено и в объединение.

Предельность. Если каждый компонент является объединением множества предельных ϕ-моделей, то объединение множества компонентов также представимо в виде объединения предельных ϕ-моделей.

83. **Доказательство Теоремы 30**

1. По Лемме 53, $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ является ϕ-моделью.

2. По Лемме 52, $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ конформна и мажорантна.

3. Из предыдущего следует, что выполнены достаточные условия монотонности 6 и 7. По Теореме 24, Теореме 27, Теореме 28 выполнены достаточные условия монотонности 1,2,3. По Теореме 29, выполнены достаточные условия монотонности 4,5 и 8. Поэтому, по Теореме 23, ϕ-модель $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ (лево-)монотонна.

4. Покажем, что ϕ-модель $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ предельна.

Рассмотрим LTS $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$.

По Лемме 44, множество конечных ϕ-трасс ϕ-модели $\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ совпадает с множеством конечных ϕ-трасс LTS $\boldsymbol{\Phi 2L}\circ\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$.

По Лемме 46, LTS $\boldsymbol{\Phi 2L}\circ\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ предельная.

Нам достаточно показать, что ϕ-модель $\boldsymbol{\Phi}^{\omega}\circ\boldsymbol{\Phi 2L}\circ\boldsymbol{Conf}_{\mathfrak{R}}(\boldsymbol{\Sigma})$ конформна, то есть добавление предела бесконечно возрастающей последовательности конформных ϕ-трасс не делает модель неконформной.

Это очевидным образом следует из того, что конформность определяется на конечных ***F***-трассах.

84. **Доказательство Теоремы 31**

1. Сначала покажем, что α-мажорирование ϕ-трасс является частичным порядком. Для этого достаточно показать рефлексивность, транзитивность и антисимметричность мажорирования символов.

1.1. Мажорирование базовых символов рефлексивно, транзитивно и антисимметрично, поскольку совпадает с равенством: $a \preccurlyeq b = a=b$.

1.2. Мажорирование ϕ-символов рефлексивно, поскольку

$a \preccurlyeq a \quad \Leftrightarrow \quad a_r \subseteq a_r \cup a_g \ \& \ a_g \subseteq a_g;$

транзитивно, поскольку

$a \preccurlyeq b \preccurlyeq c \quad \Leftrightarrow \quad a_r \subseteq b_r \cup b_g \ \& \ a_g \subseteq b_g \ \& \ b_r \subseteq c_r \cup c_g \ \& \ b_g \subseteq c_g$

$\Rightarrow \quad a_r \subseteq c_r \cup c_g \cup c_g \ \& \ a_g \subseteq c_g$

$\Rightarrow \quad a_r \subseteq c_r \cup c_g \ \& \ a_g \subseteq c_g$

$\Rightarrow \quad a \preccurlyeq c;$

антисимметрично, поскольку

$a \preccurlyeq b \preccurlyeq a \quad \Leftrightarrow \quad a_r \subseteq b_r \cup b_g \ \& \ a_g \subseteq b_g \ \& \ b_r \subseteq a_r \cup a_g \ \& \ b_g \subseteq a_g$

$\Rightarrow \quad a_r \subseteq b_r \cup a_g \ \& \ a_g = b_g \ \& \ b_r \subseteq a_r \cup b_g$

$\Rightarrow \quad a_r \subseteq b_r \ \& \ a_g = b_g \ \& \ b_r \subseteq a_r$, так как $a_r \cap a_g = b_r \cap b_g = \varnothing$,

$\Rightarrow \quad a_r = b_r \ \& \ a_g = b_g$

$\Rightarrow \quad a = b.$

2. Теперь покажем, что ϕ-мажорирование с учётом γ-мажорирования также является частичным порядком.

2.1. Рефлексивность. Поскольку, по 1, $\boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\mu}$, имеем $\boldsymbol{\mu} \preccurlyeq \boldsymbol{\mu}$.

2.2. Транзитивность. В силу 1, достаточно рассмотреть случаи с участием γ-мажорирования.

2.2.1. $\boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\lambda} \preccurlyeq^{\gamma} \boldsymbol{\sigma}$

$\Rightarrow \ \exists \boldsymbol{\lambda}_1 \leq \boldsymbol{\lambda}, \boldsymbol{\sigma}_1 \ \ \boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\lambda} \ \& \ \boldsymbol{\lambda}_1 \preccurlyeq^{\alpha} \boldsymbol{\sigma}_1 \ \& \ \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow \ \exists \boldsymbol{\mu}_1 \leq \boldsymbol{\mu}, \boldsymbol{\lambda}_1, \boldsymbol{\sigma}_1 \ \ \boldsymbol{\mu}_1 \preccurlyeq^{\alpha} \boldsymbol{\lambda}_1 \ \& \ \boldsymbol{\lambda}_1 \preccurlyeq^{\alpha} \boldsymbol{\sigma}_1 \ \& \ \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow \ \exists \boldsymbol{\mu}_1 \leq \boldsymbol{\mu}, \boldsymbol{\sigma}_1 \ \ \boldsymbol{\mu}_1 \preccurlyeq^{\alpha} \boldsymbol{\sigma}_1 \ \& \ \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\boldsymbol{\mu} \succ^{\gamma} \boldsymbol{\sigma}$.

2.2.2. $\boldsymbol{\mu} \succ^{\gamma} \boldsymbol{\lambda} \succ^{\alpha} \boldsymbol{\sigma}$

$\Rightarrow$ $\exists \boldsymbol{\mu}_1 \leq \boldsymbol{\mu}, \boldsymbol{\lambda}_1 \;\; \boldsymbol{\mu}_1 \succ^{\alpha} \boldsymbol{\lambda}_1 \;\; \& \;\; \boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \cdot \langle \gamma \rangle \;\; \& \;\; \boldsymbol{\lambda} \succ^{\alpha} \boldsymbol{\sigma}$

$\Rightarrow$ $\exists \boldsymbol{\mu}_1 \leq \boldsymbol{\mu}, \boldsymbol{\lambda}_1, \boldsymbol{\sigma}_1 \;\; \boldsymbol{\mu}_1 \succ^{\alpha} \boldsymbol{\lambda}_1 \;\; \& \;\; \boldsymbol{\lambda}_1 \succ^{\alpha} \boldsymbol{\sigma}_1 \;\; \& \;\; \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\exists \boldsymbol{\mu}_1 \leq \boldsymbol{\mu}, \boldsymbol{\sigma}_1 \;\; \boldsymbol{\mu}_1 \succ^{\alpha} \boldsymbol{\sigma}_1 \;\; \& \;\; \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\boldsymbol{\mu} \succ^{\gamma} \boldsymbol{\sigma}$.

2.2.3. $\boldsymbol{\mu} \succ^{\gamma} \boldsymbol{\lambda} \succ^{\gamma} \boldsymbol{\sigma}$

$\Rightarrow$ $\exists \boldsymbol{\mu}_1 \leq \boldsymbol{\mu}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \leq \boldsymbol{\lambda}, \boldsymbol{\sigma}_1 \;\; \boldsymbol{\mu}_1 \succ^{\alpha} \boldsymbol{\lambda}_1 \;\; \& \;\; \boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \cdot \langle \gamma \rangle \;\; \& \;\; \boldsymbol{\lambda}_2 \succ^{\alpha} \boldsymbol{\sigma}_1 \;\; \& \;\; \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\exists \boldsymbol{\mu}_2 \leq \boldsymbol{\mu}, \boldsymbol{\lambda}_2, \boldsymbol{\sigma}_1 \;\; \boldsymbol{\mu}_2 \succ^{\alpha} \boldsymbol{\lambda}_2 \;\; \& \;\; \boldsymbol{\lambda}_2 \succ^{\alpha} \boldsymbol{\sigma}_1 \;\; \& \;\; \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\exists \boldsymbol{\mu}_2 \leq \boldsymbol{\mu}, \boldsymbol{\sigma}_1 \;\; \boldsymbol{\mu}_2 \succ^{\alpha} \boldsymbol{\sigma}_1 \;\; \& \;\; \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\boldsymbol{\mu} \succ^{\gamma} \boldsymbol{\sigma}$.

2.3. Антисимметричность. В силу 1, достаточно рассмотреть случаи с участием γ-мажорирования.

2.3.1. $\boldsymbol{\mu} \succ^{\alpha} \boldsymbol{\lambda} \succ^{\gamma} \boldsymbol{\mu}$

$\Rightarrow$ $\exists \boldsymbol{\lambda}_1 \leq \boldsymbol{\lambda}, \boldsymbol{\mu}_1 \;\; \boldsymbol{\mu} \succ^{\alpha} \boldsymbol{\lambda} \;\; \& \;\; \boldsymbol{\lambda}_1 \succ^{\alpha} \boldsymbol{\mu}_1 \;\; \& \;\; \boldsymbol{\mu} = \boldsymbol{\mu}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\exists \boldsymbol{\lambda}_1 \leq \boldsymbol{\lambda}, \boldsymbol{\mu}_1 \;\; \boldsymbol{\mu}_1 \succ^{\alpha} \boldsymbol{\lambda}_1 \;\; \& \;\; \boldsymbol{\lambda}_1 \succ^{\alpha} \boldsymbol{\mu}_1 \;\; \& \;\; \boldsymbol{\mu} = \boldsymbol{\mu}_1 \cdot \langle \gamma \rangle \;\; \& \;\; \boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\boldsymbol{\mu}_1 = \boldsymbol{\lambda}_1 \;\; \& \;\; \boldsymbol{\mu} = \boldsymbol{\mu}_1 \cdot \langle \gamma \rangle \;\; \& \;\; \boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \cdot \langle \gamma \rangle$

$\Rightarrow$ $\boldsymbol{\mu} = \boldsymbol{\lambda}$.

2.3.2. $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\lambda}\preccurlyeq^{\alpha}\boldsymbol{\mu} \Rightarrow \boldsymbol{\lambda}\preccurlyeq^{\alpha}\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\lambda}$, что отличается от предыдущего случая только именованием ϕ-трасс.

2.3.3. $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\lambda}\preccurlyeq^{\gamma}\boldsymbol{\mu}$

$\Rightarrow \exists\boldsymbol{\mu}_1\leq\boldsymbol{\mu}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2\leq\boldsymbol{\lambda}, \boldsymbol{\mu}_2 \ \boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\lambda}_1 \ \& \ \boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\langle\gamma\rangle \ \& \ \boldsymbol{\lambda}_2\preccurlyeq^{\alpha}\boldsymbol{\mu}_2 \ \& \ \boldsymbol{\mu}=\boldsymbol{\mu}_2\cdot\langle\gamma\rangle$

$\Rightarrow \boldsymbol{\mu}_1=\boldsymbol{\mu}_2 \ \& \ \boldsymbol{\lambda}_2=\boldsymbol{\lambda}_1 \ \& \ \boldsymbol{\lambda}_2\preccurlyeq^{\alpha}\boldsymbol{\mu}_2 \ \& \ \boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\lambda}_1 \ \& \ \boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\langle\gamma\rangle \ \& \ \boldsymbol{\mu}=\boldsymbol{\mu}_2\cdot\langle\gamma\rangle$

$\Rightarrow \boldsymbol{\mu}_1=\boldsymbol{\mu}_2=\boldsymbol{\lambda}_2=\boldsymbol{\lambda}_1 \ \& \ \boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\langle\gamma\rangle \ \& \ \boldsymbol{\mu}=\boldsymbol{\mu}_2\cdot\langle\gamma\rangle$

$\Rightarrow \boldsymbol{\mu}=\boldsymbol{\lambda}$.

3. Рефлексивность ϕ-мажорирования «многие ко многим».

$\boldsymbol{\Lambda}\preccurlyeq\boldsymbol{\Lambda}$, поскольку для каждой ϕ-трассы $\boldsymbol{\lambda}\in\boldsymbol{\Lambda}$ имеет место $\boldsymbol{\lambda}\preccurlyeq\boldsymbol{\lambda}$.

4. Транзитивность ϕ-мажорирования «многие ко многим».

Если $\boldsymbol{\Lambda}\preccurlyeq\boldsymbol{M}$ и $\boldsymbol{M}\preccurlyeq\boldsymbol{\Sigma}$, то для каждой ϕ-трассы $\boldsymbol{\lambda}\in\boldsymbol{\Lambda}$ найдётся ϕ-трасса $\boldsymbol{\mu}\in\boldsymbol{M}$ такая, что $\boldsymbol{\lambda}\preccurlyeq\boldsymbol{\mu}$, а тогда найдётся такая ϕ-трасса $\boldsymbol{\sigma}\in\boldsymbol{\Sigma}$ такая, что $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$. В силу транзитивности ϕ-мажорирования «один к одному», $\boldsymbol{\lambda}\preccurlyeq\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$ влечёт $\boldsymbol{\lambda}\preccurlyeq\boldsymbol{\sigma}$, что и требовалось доказать.

85. Доказательство Теоремы 32

Конечная ϕ-трасса мажорируется только конечной ϕ-трассой, а бесконечная ϕ-трасса порождает пустое множество ***F***-трасс, которое мажорируется любым множеством ***F***-трасс. Поэтому нам достаточно рассмотреть случай конечных ϕ-трасс:

$\boldsymbol{\Phi}(\mathtt{I})\preccurlyeq\boldsymbol{\Phi}(\mathtt{S}) \Rightarrow \cup\circ\xi\circ\boldsymbol{\Phi}(\mathtt{I})\preccurlyeq_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}(\mathtt{S})$.

Поскольку $\mathfrak{R}$-мажорирование ***F***-трасс, порождаемых некоторыми ϕ-трассами, означает $\mathfrak{R}$-мажорирование их $\mathfrak{R}$-проекций, которые также порождаются этими ϕ-трассами, нам достаточно показать следующее:

Пусть $\boldsymbol{\mu}\in\boldsymbol{\Phi}(\mathtt{I})$, $\mu\in\xi(\boldsymbol{\mu})$ и $\mu\in \mathrm{L}_{\mathfrak{R}\Delta\gamma}$, тогда $\mu\preccurlyeq_{\mathfrak{R}}\cup_{\circ}\xi_{\circ}\boldsymbol{\Phi}(\mathtt{S})$.

Для ϕ-трассы $\boldsymbol{\mu}$ должна найтись мажоранта $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}\in\boldsymbol{\Phi}(\mathtt{S})$.

Рассмотрим возможные виды ϕ-мажорирования.

1. $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$.

Каждый $\mathfrak{R}$-отказ $\mathtt{P}$ в $\mathfrak{R}$-трассе μ вложен в ***ref***-множество соответствующего ϕ-символа $\mathtt{m}$ из порождающей ϕ-трассы $\boldsymbol{\mu}$: $\mathtt{P}\subseteq \mathtt{m}_{\mathrm{r}}$.

Поскольку $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$, для соответствующего ϕ-символа $\mathtt{s}$ из ϕ-мажоранты $\boldsymbol{\sigma}$ имеет место $\mathtt{m}_{\mathrm{r}}\subseteq \mathtt{s}_{\mathrm{r}}\cup \mathtt{s}_{\mathrm{g}}$. Следовательно, $\mathtt{P}\subseteq \mathtt{s}_{\mathrm{r}}\cup \mathtt{s}_{\mathrm{g}}$.

Здесь возможны два случая.

1.1. Для каждого $\mathfrak{R}$-отказа $\mathtt{P}$ в $\mathfrak{R}$-трассе μ имеет место $\mathtt{P}\subseteq \mathtt{s}_{\mathrm{r}}$. Покажем, что $\mu\in\cup_{\circ}\xi_{\circ}\boldsymbol{\Phi}(\mathtt{S})$.

Действительно, в данном случае, поскольку ϕ-трассы $\boldsymbol{\mu}$ и $\boldsymbol{\sigma}$, а также $\mathfrak{R}$-трасса μ, имеют одинаковые подтрассы базовых символов, имеем: $\mu\in\xi(\boldsymbol{\sigma})\subseteq\cup_{\circ}\xi_{\circ}\boldsymbol{\Phi}(\mathtt{S})$.

1.2. Для некоторого $\mathfrak{R}$-отказа $\mathtt{P}$ в $\mathfrak{R}$-трассе μ имеет место $\mathtt{P}\subseteq \mathtt{s}_{\mathrm{r}}\cup \mathtt{s}_{\mathrm{g}}$, но $\mathtt{P}\not\subseteq \mathtt{s}_{\mathrm{r}}$. Покажем, что $\mu\preccurlyeq_{\mathfrak{R}}^{\mathtt{P}}\cup_{\circ}\xi_{\circ}\boldsymbol{\Phi}(\mathtt{S})$.

Выберем первый такой $\mathfrak{R}$-отказ $\mathtt{P}$ в $\mathfrak{R}$-трассе μ: $\mu=\mu`\cdot\langle \mathtt{P}\rangle\cdot\lambda$.

Префикс $\mu`$ порождается префиксом ϕ-трассы $\boldsymbol{\mu}`\cdot\langle m\rangle\leq\boldsymbol{\mu}$, который α-мажорируется префиксом $\boldsymbol{\sigma}`\cdot\langle s\rangle\leq\boldsymbol{\sigma}$, причём m и s ϕ-символы, $P\subseteq s_r\cup s_g$, но $P\not\subseteq s_r$.

Тогда, по доказанному случаю 1.1, $\mu`\in\xi(\boldsymbol{\sigma}`\cdot\langle s\rangle)$.

Также найдётся такое внешнее действие $z\in P\cap s_g$.

Тогда найдётся ϕ-трасса $\boldsymbol{\sigma}`\cdot\langle s\rangle\cdot\langle z,\gamma\rangle\in\boldsymbol{\Phi}(\boldsymbol{s})$.

Следовательно, имеется $\mathfrak{R}$-трасса $\mu`\cdot\langle z,\gamma\rangle\in\xi(\boldsymbol{\sigma}`\cdot\langle s\rangle\cdot\langle z,\gamma\rangle)$.

Очевидно, $\xi(\boldsymbol{\sigma}`\cdot\langle s\rangle\cdot\langle z,\gamma\rangle)\subseteq\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$.

Мы имеем $\mu\preccurlyeq^{P}_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$.

2. У ϕ-трассы $\boldsymbol{\mu}$ нет α-мажоранты в $\boldsymbol{\Phi}(\boldsymbol{s})$ и $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\sigma}$.

Тогда $\exists\boldsymbol{\mu}_1<\boldsymbol{\mu}\ \exists\boldsymbol{\sigma}_1\ \boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\sigma}_1\ \&\ \boldsymbol{\sigma}=\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle$.

Тогда найдётся префикс $\mu_1<\mu$, который порождается префиксом $\boldsymbol{\mu}_1$.

По доказанному, 1.1 $\mu_1\in\xi(\boldsymbol{\sigma}_1)$ или 1.2 $\mu_1\preccurlyeq^{P}_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$.

В случае 1.1 $\mu_1\cdot\langle\gamma\rangle\in\xi(\boldsymbol{\sigma})$.

А тогда $\langle\gamma\rangle\in\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$, если $\mu_1=\epsilon$, или $\mu\preccurlyeq^{z}_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$, если $\mu_1\neq\epsilon$, поскольку разрушение может следовать только после внешнего действия.

В любом варианте $\mu\preccurlyeq_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$.

В случае 1.2 также $\mu\preccurlyeq^{P}_{\mathfrak{R}}\cup\circ\xi\circ\boldsymbol{\Phi}(\boldsymbol{s})$.

Теорема доказана.

86. **Доказательство Теоремы 33**

Пусть $A,B\subseteq Z^{\#}$, $\mathbf{I}_1,\mathbf{S}_1\in \boldsymbol{LTS}(A_\gamma)$, $\mathbf{I}_2\in \boldsymbol{LTS}(B_\gamma)$.

Обозначим $\mathbf{I_1}=\mathbf{\Phi}^{\omega}(\mathbf{I}_1)$, $\mathbf{I_2}=\mathbf{\Phi}^{\omega}(\mathbf{I}_2)$, $\mathbf{\Sigma_1}=\mathbf{\Phi}^{\omega}(\mathbf{S}_1)$.

Пусть $\boldsymbol{\kappa}\in\mathbf{I_1}$, $\boldsymbol{\lambda}\in\mathbf{I_2}$, $\boldsymbol{\kappa}`\in\mathbf{\Sigma_1}$ и $\boldsymbol{\kappa}\preccurlyeq\boldsymbol{\kappa}`$.

Возьмём произвольную ϕ-трассу $\boldsymbol{\mu}\in\boldsymbol{\kappa}\wr\boldsymbol{\lambda}$.

Нам надо показать, что $\boldsymbol{\mu}\preccurlyeq\cup(\mathbf{\Sigma_1}\wr\mathbf{I_2})$.

Рассмотрим два возможных случая ϕ-мажорирования.

1. $\boldsymbol{\kappa}\preccurlyeq^{\alpha}\boldsymbol{\kappa}`$.

Для данных ϕ-трасс $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$ ϕ-трасса $\boldsymbol{\mu}$ однозначно определяется последовательностью применения правил композиции $(\wr_\phi 1\div 4)$ и, быть может, дополнительным правилом $(\wr_\phi 5)$ для построения ϕ-трасс, завершающихся Δ-символом, моделирующим синхронную дивергенцию, или правилом $(\wr_\phi 6)$ для построения бесконечных ϕ-трасс.

Из определения α-мажорирования следует, что ϕ-трассы $\boldsymbol{\kappa}$ и $\boldsymbol{\kappa}`$ отличаются только $i$-ыми ϕ-символами $a=\boldsymbol{\kappa}^{\downarrow}\Phi(A)(i)$ и $a`=\boldsymbol{\kappa}`^{\downarrow}\Phi(A)(i)$. Отсюда следует, что для любой последовательности применения правил композиции ϕ-трасс $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$, определяющей ϕ-трассу $\boldsymbol{\mu}$, эту последовательность можно применить к ϕ-трассам $\boldsymbol{\kappa}`$ и $\boldsymbol{\lambda}$, и получить некоторую ϕ-трассу $\boldsymbol{\mu}`$. При этом ϕ-трассы $\boldsymbol{\mu}$ и $\boldsymbol{\mu}`$ могут отличаться только результатами композиций ϕ-символов: для $b=\boldsymbol{\lambda}^{\downarrow}\Phi(B)(i)$ могут различаться $a\wr b$ и $a`\wr b$.

Рассмотрим два возможных варианта.

1.1. 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 $\boldsymbol{\kappa}`_0$ и $\boldsymbol{\lambda}_0$, которые получаются из ϕ-трасс $\boldsymbol{\kappa}`$ и $\boldsymbol{\lambda}$, соответственно, удалением некоторых ϕ-символов, и такие, что некоторая ϕ-трасса $\boldsymbol{\mu}`_0 \in \boldsymbol{\kappa}`_0 \between \boldsymbol{\lambda}_0$ α-мажорирует ϕ-трассу $\boldsymbol{\mu}$: $\boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\mu}`_0$.

Отсюда будет следовать нужное утверждение. Действительно, по замкнутости ϕ-модели по ***d***-операции, $\boldsymbol{\kappa}`_0 \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_0 \in \boldsymbol{\Pi_2}$, что влечёт $\boldsymbol{\mu}`_0 \in \boldsymbol{\kappa}`_0 \between \boldsymbol{\lambda}_0 \subseteq \cup(\boldsymbol{\Sigma_1} \between \boldsymbol{\Pi_2})$. Тем самым, $\boldsymbol{\mu} \preccurlyeq \cup(\boldsymbol{\Sigma_1} \between \boldsymbol{\Pi_2})$.

Нам достаточно показать, что $a \between b \neq \tau$ влечёт $a` \between b \neq \tau$ и $a \between b \preccurlyeq a` \between b$.

Действительно, если это условие выполнено, то мажорирование $\boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\mu}`$ может быть нарушено только из-за того, что при построении ϕ-трассы $\boldsymbol{\mu}$ композиция ϕ-символов $a \between b = \tau$, а при построении ϕ-трассы $\boldsymbol{\mu}`$ композиция ϕ-символов $a` \between b \neq \tau$.

Удаление всех таких ϕ-символов $a`$ из ϕ-трассы $\boldsymbol{\kappa}`$ и соответствующих им ϕ-символов b из ϕ-трассы $\boldsymbol{\lambda}$ 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 $\boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\mu}`_0$.

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, $(A \setminus a_r) \cap (\underline{B \setminus b_r}) = \varnothing$.

По определению мажорирования ϕ-трасс, $a_r \subseteq a`_r \cup a`_g$ и $a_g \subseteq a`_g$.

Покажем, что $(A \setminus a`_r) \cap (\underline{B \setminus b_r}) = \varnothing$.

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 $z \notin (A \setminus a`_r) \cap (\underline{B \setminus b_r})$.

ii. $z \in A \cap \underline{B}$ и $z \in a`_r$.

Тогда $z \notin A \setminus a`_r$ и, следовательно, $z \notin (A \setminus a`_r) \cap (\underline{B \setminus b_r})$.

iii. $z \in A \cap \underline{B}$ и $z \in a`_g$.

Тогда, по условию рассматриваемого варианта a, $\underline{z} \in b_r$, что влечёт $z \notin \underline{B \setminus b_r}$ и, следовательно, $z \notin (A \setminus a`_r) \cap (\underline{B \setminus b_r})$.

iv. $z \in A \cap \underline{B}$, $z \notin a`_r$ 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, $z \in A \setminus a_r$. Отсюда, поскольку $(A \setminus a_r) \cap (\underline{B \setminus b_r}) = \varnothing$, имеем $z \notin \underline{B \setminus b_r}$, что влечёт $z \notin (A \setminus a`_r) \cap (\underline{B \setminus b_r})$.

Мы доказали, что $(A \setminus a`_r) \cap (\underline{B \setminus b_r}) = \varnothing$.

Отсюда, по определению композиции ϕ-символов, $a`⌋\!\lfloor b \neq \tau$.

Теперь покажем, что $a⌋\!\lfloor b \preccurlyeq a`⌋\!\lfloor b$.

По определению композиции ϕ-символов,

во-первых, $(a⌋\!\lfloor b)_r = (A \setminus \underline{B}) \cap a_r \cup (B \setminus \underline{A}) \cap b_r$

и $(a`⌋\!\lfloor b)_r = (A \setminus \underline{B}) \cap a`_r \cup (B \setminus \underline{A}) \cap b_r$,

во-вторых, $(a⌋\!\lfloor b)_g = (A \setminus \underline{B}) \cap a_g \cup (B \setminus \underline{A}) \cap b_g$

и $(a`⌋\!\lfloor b)_g = (A \setminus \underline{B}) \cap a`_g \cup (B \setminus \underline{A}) \cap b_g$.

Поскольку $a_r \subseteq a`_r \cup a`_g$, имеем $(a⌋\!\lfloor b)_r \subseteq (a`⌋\!\lfloor b)_r \cup (a`⌋\!\lfloor b)_g$.

Поскольку, $a_g \subseteq a`_g$, имеем $(a⌋\!\lfloor b)_g \subseteq (a`⌋\!\lfloor b)_g$.

Отсюда, по определению мажорирования ϕ-символов, $a⌉⌊b \preccurlyeq a`⌉⌊b$.

Утверждение Теоремы для варианта 1.1 доказано.

1.2. 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 $\boldsymbol{\mu} \preccurlyeq^{\gamma} \cup (\boldsymbol{\Sigma_1}⌉⌊\boldsymbol{\Gamma_2})$.

Выберем самый первый такой индекс i.

Тогда ϕ-трассы можно представить в таком виде:

$\boldsymbol{\kappa}=\boldsymbol{\kappa}_1 \cdot \langle a \rangle \cdot \boldsymbol{\kappa}_2$, $\boldsymbol{\kappa}`=\boldsymbol{\kappa}`_1 \cdot \langle a` \rangle \cdot \boldsymbol{\kappa}`_2$, $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1 \cdot \langle b \rangle \cdot \boldsymbol{\lambda}_2$,

$\boldsymbol{\mu}=\boldsymbol{\mu}_1 \cdot (\langle a⌉⌊b \rangle \uparrow \{\tau\}) \cdot \boldsymbol{\mu}_2$, $\boldsymbol{\mu}`=\boldsymbol{\mu}`_1 \cdot (\langle a`⌉⌊b \rangle \uparrow \{\tau\}) \cdot \boldsymbol{\mu}`_2$,

$\boldsymbol{\mu}_1 \in \boldsymbol{\kappa}_1⌉⌊\boldsymbol{\lambda}_1$, $\boldsymbol{\mu}`_1 \in \boldsymbol{\kappa}`_1⌉⌊\boldsymbol{\lambda}_1$.

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 префикс-замкнутости ϕ-модели, $\boldsymbol{\kappa}_1 \cdot \langle a \rangle \in \boldsymbol{\Gamma_1}$, $\boldsymbol{\kappa}`_1 \cdot \langle a` \rangle \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_1 \cdot \langle b \rangle \in \boldsymbol{\Gamma_2}$.

Тогда, по доказанному утверждению в варианте 1.1, существуют ϕ-трассы $\boldsymbol{\kappa}`_{10} \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_{10} \in \boldsymbol{\Gamma_2}$, 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 замкнутости ϕ-модели по ***d***-операции, $\boldsymbol{\kappa}`_{10} \cdot \langle a` \rangle \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_{10} \cdot \langle b \rangle \in \boldsymbol{\Gamma_2}$.

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 полноте ϕ-модели, $\boldsymbol{\kappa}`_{10} \cdot \langle z, \gamma \rangle \in \boldsymbol{\Sigma_1}$.

Поскольку, по условию рассматриваемого варианта 1.2, $\underline{z}\in A\cap\underline{B}\setminus b_r$ и $\boldsymbol{\lambda}_{10}\cdot\langle b\rangle\in\mathbf{I_2}$, то, по полноте ϕ-модели, $\boldsymbol{\lambda}_{10}\cdot\langle\underline{z}\rangle\in\mathbf{I_2}$.

Следовательно, $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in\cup(\boldsymbol{\Sigma_1}\rceil\lfloor\mathbf{I_2})$.

По определению мажорирования ϕ-трасс, $\boldsymbol{\mu}_1\leq\boldsymbol{\mu}$ и $\boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\mu}`_{10}$ влечёт $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle$.

Итак, $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle$ и $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in\cup(\boldsymbol{\Sigma_1}\rceil\lfloor\mathbf{I_2})$, что влечёт $\boldsymbol{\mu}\preccurlyeq^{\gamma}\cup(\boldsymbol{\Sigma_1}\rceil\lfloor\mathbf{I_2})$.

Утверждение 1.2 доказано.

Для случая 1 Теорема доказана.

2. У ϕ-трассы $\boldsymbol{\kappa}$ нет α-мажоранты в $\boldsymbol{\Sigma_1}$ и $\boldsymbol{\kappa}\preccurlyeq^{\gamma}\boldsymbol{\kappa}`$.

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 $\boldsymbol{\kappa}_2\notin\{\epsilon,\langle\gamma\rangle\}$, $\boldsymbol{\kappa}_1\preccurlyeq^{\alpha}\boldsymbol{\kappa}`_1$, $\boldsymbol{\mu}_1\in\boldsymbol{\kappa}_1\rceil\lfloor\boldsymbol{\lambda}_1$.

Здесь последовательность правил $(\rceil_{\phi}\lfloor 1\div 4)$, порождающих $\boldsymbol{\mu}_1$, является префиксом последовательности правил $(\rceil_{\phi}\lfloor 1\div 4)$, порождающих $\boldsymbol{\mu}$.

По доказанному случаю α-мажорирования, возможна два варианта.

2.1. Существуют такие ϕ-трассы $\boldsymbol{\kappa}`_{10}\in\boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_{10}\in\mathbf{I_2}$, которые получаются из ϕ-трасс $\boldsymbol{\kappa}`_1$ и $\boldsymbol{\lambda}_1$, соответственно, удалением некоторых ϕ-символов, и такие, что некоторая ϕ-трасса $\boldsymbol{\mu}`_{10}\in\boldsymbol{\kappa}`_{10}\rceil\lfloor\boldsymbol{\lambda}_{10}$ мажорирует $\boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\mu}`_{10}$.

По правилу вывода $(\rceil_{\phi}\lfloor 1)$, $\boldsymbol{\mu}`_{10}\in\boldsymbol{\kappa}`_{10}\rceil\lfloor\boldsymbol{\lambda}_{10}$ влечёт $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in(\boldsymbol{\kappa}`_{10}\cdot\langle\gamma\rangle)\rceil\lfloor\boldsymbol{\lambda}_{10}$.

По замкнутости ϕ-модели по ***d***-операции, $\boldsymbol{\kappa}`_{10}\cdot\langle\gamma\rangle\in\boldsymbol{\Sigma_1}$.

Тем самым, $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in(\boldsymbol{\kappa}`_{10}\cdot\langle\gamma\rangle)\rceil\lfloor\boldsymbol{\lambda}_{10}\subseteq\cup(\boldsymbol{\Sigma_1}\rceil\lfloor\mathbf{I_2})$.

Следовательно, $\boldsymbol{\mu} \preccurlyeq^{\gamma} \cup(\boldsymbol{\Sigma}_1 \upharpoonright\!\downharpoonright \mathbf{I}_2)$.

2.2. $\boldsymbol{\mu}_1 \preccurlyeq^{\gamma} \cup(\boldsymbol{\Sigma}_1 \upharpoonright\!\downharpoonright \mathbf{I}_2)$.

Тогда $\boldsymbol{\mu}_1 \leq \boldsymbol{\mu}$ влечёт $\boldsymbol{\mu} \preccurlyeq^{\gamma} \cup(\boldsymbol{\Sigma}_1 \upharpoonright\!\downharpoonright \mathbf{I}_2)$.

Для случая 2 утверждение доказано.

Утверждение доказано.

87. **Доказательство Теоремы 34**

Пусть для $\mathbf{I}_1, \mathbf{I}_2, \mathbf{S}_1, \mathbf{S}_2 \in \boldsymbol{LTS}_{\gamma}$ имеет место ϕ-мажорирование:

$\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1) \preccurlyeq \boldsymbol{\Phi}^{\omega}(\mathbf{S}_1)$ & $\boldsymbol{\Phi}^{\omega}(\mathbf{I}_2) \preccurlyeq \boldsymbol{\Phi}^{\omega}(\mathbf{S}_2)$.

По композиционности ϕ-мажорирования слева (Теорема 33),

$\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{I}_2)) \preccurlyeq \cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{I}_2))$.

По коммутативности композиции ϕ-трасс (Лемма 50),

$\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{I}_2)) = \cup(\boldsymbol{\Phi}^{\omega}(\mathbf{I}_2) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{S}_1))$.

По композиционности ϕ-мажорирования слева (Теорема 33),

$\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{I}_2) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{S}_1)) \preccurlyeq \cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_2) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{S}_1))$.

По коммутативности композиции ϕ-трасс (Лемма 50),

$\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_2) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{S}_1)) = \cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{S}_2))$.

По транзитивности ϕ-мажорирования (Теорема 31),

$\cup(\boldsymbol{\Phi}^{\omega}(\mathbf{I}_1) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{I}_2)) \preccurlyeq \cup(\boldsymbol{\Phi}^{\omega}(\mathbf{S}_1) \upharpoonright\!\downharpoonright \boldsymbol{\Phi}^{\omega}(\mathbf{S}_2))$.

88. **Доказательство Теоремы 35**

Выше доказано выполнение условий монотонности 1÷5 и 8 (Теорема 24, Теорема 27, Теорема 28, Теорема 32, Теорема 34 и Теорема 31).

Поэтому для монотонности ϕ-модели требуется выполнение условия 6 – конформность (исходной спецификации), и условия 7 – мажорантность (мажорирование конформных ϕ-трасс).

Любая подмодель монотонной ϕ-модели конформна. Действительно, по генеративности ϕ-трасс (Теорема 27), вложенность ϕ-моделей влечёт вложенность соответствующих $\mathfrak{R}$-моделей. По Лемме 16, если первая $\mathfrak{R}$-модель вложена во вторую $\mathfrak{R}$-модель, то первая $\mathfrak{R}$-модель конформна второй $\mathfrak{R}$-модели. По определению, монотонная ϕ-модель конформна спецификации. Поэтому, по транзитивности конформности (Теорема 17), подмодель монотонной ϕ-модели конформна.

Поэтому, по Теореме 23, для монотонности достаточно, чтобы подмодель мажорировала все конформные ϕ-трассы.

89. **Доказательство Леммы 54**

1. Если в спецификации есть γ-трасса, то финальными трассами являются только пустая ϕ-трасса ϵ и γ-трасса $\langle\gamma\rangle$. Эти ϕ-трассы конформны, поскольку ϕ-модель, состоящая только из этих ϕ-трасс, конформна.

Далее будем считать, что в спецификации нет γ-трассы.

2. Безопасная ϕ-трасса $\boldsymbol{\sigma}=\boldsymbol{\mu}$ конформна по определению.

3. Пусть ϕ-трасса $\boldsymbol{\mu}$ безопасна, $\boldsymbol{postf}(\boldsymbol{\mu})=\epsilon$, а все $\mathfrak{R}$-кнопки опасны после $\boldsymbol{\mu}$.

Покажем, что ϕ-трасса $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ конформна.

Пусть $\mathbf{T}$ одна из предельных ϕ-моделей, объединению которых равна ϕ-модель $\boldsymbol{Conf}(\boldsymbol{\Sigma})$, и такая, что $\boldsymbol{\mu}\in\mathbf{T}$. Тогда в ϕ-модель $\mathbf{T}$ добавим множество ϕ-трасс $\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle\Delta\rangle$: $\mathbf{T}`=\mathbf{T}\cup\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle\Delta\rangle$.

3.1. Покажем, что $\mathbf{T`}$ предельная ϕ-модель.

Поскольку ϕ-трасса $\boldsymbol{\mu}$ допустима и согласована, а $\boldsymbol{postf}(\boldsymbol{\mu})=\epsilon$, все добавляемые ϕ-трассы допустимы и согласованы.

Конвергентность также не нарушается при добавлении таких ϕ-трасс (ϕ-символ должен продолжать только такую ϕ-трассу, которая не заканчивается и не продолжается дивергенцией).

Сохранение замкнутости гарантируется добавлением ***dr***-замкнутого множества ϕ-трасс.

Полнота сохраняется, так как мы не добавляем новых ϕ-символов и новых внешних действий (если $\boldsymbol{\mu}\in\mathbf{T}$, то, по замкнутости ϕ-модели $\mathbf{T}$, $\boldsymbol{dr}(\boldsymbol{\mu})\subseteq\mathbf{T}$).

Предельность сохраняется, так как мы не добавили ни одной бесконечно возрастающей цепочки ϕ-трасс.

3.2. Покажем, что ϕ-модель $\mathbf{T`}$ конформна. Действительно, ϕ-модель $\mathbf{T}$ конформна. В результате добавления ϕ-трасс $\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle\Delta\rangle$ нарушиться может только гипотеза о безопасности и только в том случае, когда существует $\mathfrak{R}$-кнопка безопасная после $\boldsymbol{\mu}$, что не верно.

Итак, мы показали, что существует конформная ϕ-модель $\mathbf{T`}$, в которой есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle\Delta\rangle$. Следовательно, эта ϕ-трасса конформна.

4. Пусть ϕ-трасса $\boldsymbol{\mu}$ безопасна, действие $z\in L$ опасно после $\boldsymbol{\mu}$, и $\forall\circ\in\boldsymbol{Im}\circ\boldsymbol{postf}(\boldsymbol{\mu})\quad z\notin\circ_r$. Покажем, что ϕ-трассы $\boldsymbol{\mu}\cdot\langle z\rangle$ и $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$ конформны.

Пусть $\mathbf{T}$ одна из предельных ϕ-моделей, объединению которых равна ϕ-модель $\boldsymbol{Conf}(\boldsymbol{\Sigma})$, и такая, что $\boldsymbol{\mu}\in\mathbf{T}$. Тогда в ϕ-модель $\mathbf{T}$ добавим множество ϕ-трасс $\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle z\rangle\cup\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$: $\mathbf{T`}=\mathbf{T}\cup\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle z\rangle\cup\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$.

3.1. Покажем, что $\mathbf{T}'$ предельная ф-модель.

Поскольку ф-трасса $\boldsymbol{\mu}$ допустима и согласована, и $\forall \circ \in \boldsymbol{Im}_{\circ}\boldsymbol{postf}(\boldsymbol{\mu})$ $z \notin \circ_r$, все добавляемые ф-трассы допустимы и согласованы.

Конвергентность также не нарушается при добавлении таких ф-трасс (ф-символ должен продолжать только такую ф-трассу, которая не заканчивается и не продолжается разрушением).

Сохранение замкнутости гарантируется добавлением ***dr***-замкнутого множества ф-трасс.

Полнота сохраняется, так как мы не добавляем новых ф-символов и новых внешних действий (если $\boldsymbol{\mu} \in \mathbf{T}$, то, по замкнутости ф-модели $\mathbf{T}$, $\boldsymbol{dr}(\boldsymbol{\mu}) \subseteq \mathbf{T}$).

Предельность сохраняется, так как мы не добавили ни одной бесконечно возрастающей цепочки ф-трасс.

3.2. Покажем, что ф-модель $\mathbf{T}'$ конформна. Действительно, ф-модель $\mathbf{T}$ конформна. В результате добавления ф-трасс $\boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle z\rangle \cup \boldsymbol{dr}(\boldsymbol{\mu}\cdot\langle z, \gamma\rangle$ нарушиться может только гипотеза о безопасности и только в том случае, когда действие z безопасно после $\boldsymbol{\mu}$, что не верно.

Итак, мы показали, что существует конформная ф-модель $\mathbf{T}'$, в которой есть ф-трассы $\boldsymbol{\mu}\cdot\langle z\rangle$ и $\boldsymbol{\mu}\cdot\langle z, \gamma\rangle$. Следовательно, эти ф-трассы конформны.

90. Доказательство Леммы 55

1. Необходимость. Пусть конформная ф-трасса $\boldsymbol{\sigma} \in \boldsymbol{Conf}(\boldsymbol{\Sigma})$ финальна. Если $\boldsymbol{\sigma} = \epsilon$ или $\boldsymbol{\sigma} = \langle\gamma\rangle$, то утверждение очевидно. Если ф-трасса $\boldsymbol{\sigma}$ безопасна, то она не может иметь конформных γ-ответвлений, так как тогда генерируемые ею безопасные $\mathfrak{R}$-трассы имели бы в ф-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$ ответвление разрушением, что противоречит конформности этой ф-модели. Если ф-трасса

$\boldsymbol{\sigma}$ является продолжением безопасной ϕ-трассы $\boldsymbol{\mu}$, то есть $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle\Delta\rangle$, $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z\rangle$ или $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$, то γ-ответвление после префикса $\boldsymbol{\sigma}`<\boldsymbol{\sigma}$ либо было бы γ-ответвлением от безопасной ϕ-трассы $\boldsymbol{\mu}$, либо (в последнем случае) совпадало бы с самой ϕ-трассой $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$.

2. Достаточность. Пусть конформная ϕ-трасса $\boldsymbol{\sigma}\in\boldsymbol{Conf}(\boldsymbol{\Sigma})$ не имеет отличных от неё самой конформных γ-ответвлений после меньшего префикса. Допустим, ϕ-трасса $\boldsymbol{\sigma}$ не финальна.

2.1. Покажем, что пустая ϕ-трасса безопасна.

Действительно, в противном случае пустая $\mathfrak{R}$-трасса опасна, что означает $\langle\gamma\rangle\in\boldsymbol{\Sigma}$ и, следовательно, $\langle\gamma\rangle\in\boldsymbol{Conf}(\boldsymbol{\Sigma})$. А тогда ϕ-трасса $\boldsymbol{\sigma}$ либо финальна $\boldsymbol{\sigma}=\epsilon$ или $\boldsymbol{\sigma}=\langle\gamma\rangle$, либо имеет отличное от неё самой конформное γ-ответвление $\langle\gamma\rangle$ после меньшего (пустого) префикса. И то и другое противоречит допущению.

2.2. Следовательно, ϕ-трасса $\boldsymbol{\sigma}$ имеет максимальный безопасный префикс $\boldsymbol{\mu}$ и $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\boldsymbol{\lambda}$.

2.2.1. Если $\boldsymbol{\lambda}=\epsilon$, то $\boldsymbol{\sigma}$ безопасна, что противоречит её нефинальности.

2.2.2. Если $\boldsymbol{\lambda}=\langle\gamma\rangle$, то это противоречит конформности ϕ-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$.

2.2.3. Если $\boldsymbol{\lambda}=\langle\Delta\rangle$, то никакая $\mathfrak{R}$-кнопка не может быть безопасной после $\boldsymbol{\mu}$, так как это противоречило бы конформности ϕ-модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$. Но тогда ϕ-трасса $\boldsymbol{\sigma}$ финальна, что не верно.

2.2.4. Если $\boldsymbol{\lambda}$ начинается с ϕ-символа $\circ$, то ϕ-трасса $\boldsymbol{\mu}\cdot\langle\circ\rangle$ также безопасна, поскольку $\xi(\boldsymbol{\mu})\subseteq\xi(\boldsymbol{\mu}\cdot\langle\circ\rangle)$. Но это противоречит максимальности префикса $\boldsymbol{\mu}$.

2.2.5. Если $\boldsymbol{\lambda}$ начинается с внешнего действия z, то z опасен после $\boldsymbol{\mu}$, так как в противном случае ϕ-трасса $\boldsymbol{\mu}\cdot\langle z\rangle$ также безопасна, что противоречит максимальности префикса $\boldsymbol{\mu}$. А тогда $\boldsymbol{\lambda}=\langle z\rangle$, или $\boldsymbol{\lambda}=\langle z,\gamma\rangle$, или $\boldsymbol{\lambda}>\langle z\rangle$ и $\boldsymbol{\lambda}\neq\langle z,\gamma\rangle$. В первых двух случаях ϕ-трасса $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z\rangle$ или $\boldsymbol{\sigma}=\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$ финальна, что не верно. В третьем случае у ϕ-трассы $\boldsymbol{\sigma}$ есть отличное от неё самой γ-ответвление $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$ после меньшего префикса $\boldsymbol{\mu}\cdot\langle z\rangle$, чего также не должно быть.

Итак, мы во всех случаях пришли к противоречию и, следовательно, наше допущение было не верно, и ϕ-трасса $\boldsymbol{\sigma}$ финальна, что и требовалось доказать.

91. Доказательство Теоремы 36

Нам надо показать: $d\circ Final(\boldsymbol{\Sigma})\in \Phi MODEL(L)$, $d\circ Final(\boldsymbol{\Sigma})$ *saco* $\boldsymbol{\Sigma}$, $Conf(\boldsymbol{\Sigma})\preccurlyeq d\circ Final(\boldsymbol{\Sigma})$ и $d\circ Final(\boldsymbol{\Sigma})$ предельна.

1. Сначала покажем, что $d\circ Final(\boldsymbol{\Sigma})$ является предельной ϕ-моделью.

Очевидно, что ***d***-замыкание множества ϕ-трасс сохраняет все остальные свойства ϕ-модели (дерево ϕ-трасс, допустимость, согласованность, конвергентность, полноту и предельность). Поэтому нам достаточно показать, что множество финальных ϕ-трасс обладает этими остальными свойствами.

Множество финальных ϕ-трасс <u>префикс-замкнуто</u> по определению.

<u>Допустимость и согласованность</u> следуют из того, что множество финальных ϕ-трасс – это подмножество ϕ-модели $Conf(\boldsymbol{\Sigma})$.

<u>Конвергентность</u>. Если финальная ϕ-трасса $\boldsymbol{\sigma}$ не завершается и не продолжается дивергенцией и разрушением, то, по конвергентности ϕ-

модели $\boldsymbol{Conf}(\boldsymbol{\Sigma})$, найдётся такой ϕ-символ $\circ$, что $\boldsymbol{\sigma}\cdot\langle\circ\rangle\in\boldsymbol{Conf}(\boldsymbol{\Sigma})$. Если бы ϕ-трасса $\boldsymbol{\sigma}\cdot\langle\circ\rangle$ была нефинальна, то, по Лемме 55, нашёлся бы такой префикс $\boldsymbol{\mu}<\boldsymbol{\sigma}\cdot\langle\circ\rangle$, что $\boldsymbol{\mu}\cdot\langle\gamma\rangle\in\boldsymbol{Conf}(\boldsymbol{\Sigma})$. Но тогда либо $\boldsymbol{\mu}=\boldsymbol{\sigma}$, либо $\boldsymbol{\mu}<\boldsymbol{\sigma}$. В первом случае ϕ-трасса $\boldsymbol{\sigma}$ продолжается разрушением, что не верно. Во втором случае ϕ-трасса $\boldsymbol{\sigma}$ не финальна, что тоже не верно. Мы пришли к противоречию и, следовательно, ϕ-трасса $\boldsymbol{\sigma}\cdot\langle\circ\rangle$ финальна.

Полнота. Если ϕ-трасса $\boldsymbol{\sigma}\cdot\langle\circ\rangle$ финальна, то, по Лемме 55, $\forall\boldsymbol{\mu}<\boldsymbol{\sigma}\cdot\langle\circ\rangle$ $\boldsymbol{\mu}\cdot\langle\gamma\rangle\notin\boldsymbol{Conf}(\boldsymbol{\Sigma})$. Тогда для любого внешнего действия $z\notin\circ_r$ имеем $\forall\boldsymbol{\mu}<\boldsymbol{\sigma}\cdot\langle\circ,z\rangle$ $\boldsymbol{\mu}\cdot\langle\gamma\rangle\notin\boldsymbol{Conf}(\boldsymbol{\Sigma})$, то есть ϕ-трасса $\boldsymbol{\sigma}\cdot\langle\circ,z\rangle$ финальна. Очевидно, что для $z\in\circ_g$ также имеем $\forall\boldsymbol{\mu}<\boldsymbol{\sigma}\cdot\langle\circ,z,\gamma\rangle$ $\boldsymbol{\mu}\cdot\langle\gamma\rangle=\boldsymbol{\sigma}\cdot\langle\circ,z,\gamma\rangle\ \vee$ $\boldsymbol{\mu}\cdot\langle\gamma\rangle\notin\boldsymbol{Conf}(\boldsymbol{\Sigma})$, то есть ϕ-трасса $\boldsymbol{\sigma}\cdot\langle\circ,z,\gamma\rangle$ финальна.

Предельность. По Теореме 30, ϕ-модель $\boldsymbol{Conf}(\boldsymbol{\Sigma})$ предельна. Поэтому предел $\boldsymbol{\sigma}$ бесконечно возрастающей цепочки финальных ϕ-трасс $\boldsymbol{\sigma}_1<\boldsymbol{\sigma}_2<\ldots$ конформен. Если бы ϕ-трасса $\boldsymbol{\sigma}$ имела конформное γ-ответвление $\boldsymbol{\mu}\cdot\langle\gamma\rangle$, то в цепочке нашлась бы ϕ-трасса $\boldsymbol{\sigma}_i$ такая, что $\boldsymbol{\mu}<\boldsymbol{\sigma}_i$ и, очевидно, $\boldsymbol{\mu}\cdot\langle\gamma\rangle\neq\boldsymbol{\sigma}_i$. Но тогда, по Лемме 55, ϕ-трасса $\boldsymbol{\sigma}_i$ не была бы финальной, что не верно.

2. Теперь покажем, что $\boldsymbol{d}_\circ\boldsymbol{Final}(\boldsymbol{\Sigma})$ является мажорантной ϕ-моделью.

Финальная ϕ-трасса мажорирует сама себя, а для любой нефинальной ϕ-трассы $\boldsymbol{\sigma}_0$, по Лемме 55, найдётся γ-ответвление $\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle$ от её префикса $\boldsymbol{\sigma}_1<\boldsymbol{\sigma}_0$ и $\boldsymbol{\sigma}_0\neq\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle$, что означает строгое γ-мажорирование $\boldsymbol{\sigma}_0\prec\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle$. Если $\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle$ не финальна, то также найдётся строгая γ-мажоранта $\boldsymbol{\sigma}_1\cdot\langle\gamma\rangle\prec\boldsymbol{\sigma}_2\cdot\langle\gamma\rangle$. И так далее. Поскольку длина γ-мажоранты всё время уменьшается, в конце

концов мы должны получить финальную γ-мажоранту $\boldsymbol{\sigma}_n \cdot \langle \gamma \rangle$. По транзитивности ϕ-мажорирования (Теорема 31), имеем $\boldsymbol{\sigma}_0 \prec \boldsymbol{\sigma}_n \cdot \langle \gamma \rangle$. Тем самым, каждая конформная ϕ-трасса мажорируется финальной ϕ-трассой, что и требовалось доказать.

3. Теперь покажем, что $d_\circ Final(\boldsymbol{\Sigma})$ является монотонной ϕ-моделью.

По Теореме 30, наибольшая конформная ϕ-модель монотонна.

По Лемме 54, финальные ϕ-трассы конформны.

Поэтому, по Теореме 35, достаточно, чтобы $d_\circ Final(\boldsymbol{\Sigma})$ была мажорантной ϕ-моделью, что уже доказано.

92. **Доказательство Леммы 56**

Непосредственно следует из определений похожих финальных ϕ-трасс (отличаются только ***gamma***-множествами соответствующих ϕ-символов) и ξ-оператора (не зависит от ***gamma***-множеств ϕ-символов).

93. **Доказательство Леммы 57**

По Лемме 56, похожие финальные ϕ-трассы генерируют одинаковые ***F***-трассы.

Следовательно, они генерируют одинаковые $\mathfrak{R}$-трассы.

Следовательно, они генерируют одинаковые безопасные $\mathfrak{R}$-трассы.

Следовательно, действия, опасные после похожих финальных ϕ-трасс одинаковые.

Похожие ϕ-трассы отличаются только ***gamma***-множествами соответствующих ϕ-символов.

Поскольку в однородных ϕ-трассах ***gamma***-множества ϕ-символов – это множества опасных (после соответствующих префиксов) действий, похожие однородные ϕ-трассы совпадают, что и требовалось доказать.

94. **Доказательство Леммы 58**

1. Поскольку отношение похожести является эквивалентностью, то, если для финальной ϕ-трассы существует похожая однородная ϕ-трасса, то, по Лемме 57, она единственна.

2. Действия, принадлежащие ***gamma***-множеству ϕ-символу финальной ϕ-трассы $\boldsymbol{\mu}$, очевидно, опасны после префикса, завершающегося этим ϕ-символом. Поскольку ***gamma***-множество ϕ-символа однородной ϕ-трассы $\boldsymbol{\sigma}$ состоит из всех таких действий, то ***gamma***-множество каждого ϕ-символа ϕ-трассы $\boldsymbol{\mu}$ вложено в ***gamma***-множество соответствующего ϕ-символа ϕ-трассы $\boldsymbol{\sigma}$. Следовательно, $\boldsymbol{\mu}\sim\boldsymbol{\sigma}\ \&\ \boldsymbol{\sigma}\in \boldsymbol{Uniform}(\boldsymbol{\Sigma}) \Rightarrow \boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$.

3. Нам осталось показать, что для любой финальной ϕ-трассы найдётся похожая однородная ϕ-трасса. Допустим, это не так.

3.1. Пустая ϕ-трасса финальна и однородна: $\epsilon\sim\epsilon\ \&\ \epsilon\in \boldsymbol{Uniform}(\boldsymbol{\Sigma})$.

3.2. Тогда найдётся такая финальная ϕ-трасса $\boldsymbol{\mu}$, для которой есть похожая однородная ϕ-трасса $\boldsymbol{\sigma}$, а для финального продолжения ϕ-трассы $\boldsymbol{\mu}$ одним символом похожей однородной ϕ-трассы нет.

3.2.1. Сначала рассмотрим случай финального продолжения действием $u\in L$, дивергенцией $u=\Delta$ или разрушением $u=\gamma$.

Добавим в финальную ϕ-модель все ϕ-трассы вида $\boldsymbol{\sigma}\cdot\langle u\rangle\cdot\boldsymbol{\lambda}$, где $\boldsymbol{\mu}\cdot\langle u\rangle\cdot\boldsymbol{\lambda}\in \boldsymbol{Final}(\boldsymbol{\Sigma})$, а потом сделаем ***dr***-замыкание[37].

Легко проверяется, что при этом будут сохранены все свойства ϕ-модели, включая предельность. Например, это наглядно видно на канонической LTS-модели $\boldsymbol{Φ2L}\circ\boldsymbol{d}\circ\boldsymbol{Final}(\boldsymbol{\Sigma})$. Здесь добавлению ϕ-трасс с последующим ***dr***-замыканием соответствует:

---

[37] Это эквивалентно добавлению ***dr***-замыкания указанного добавляемого множества ϕ-трасс.

1) добавление перехода $\boldsymbol{\sigma}\text{—}u\rightarrow\boldsymbol{\mu}\cdot\langle u\rangle$ для $u\in L$;

2) добавление перехода $\boldsymbol{\sigma}\text{—}\tau\rightarrow\boldsymbol{\sigma}$ для $u=\Delta$;

3) добавление перехода $\boldsymbol{\sigma}\text{—}\gamma\rightarrow\boldsymbol{\sigma}$ для $u=\gamma$.

Очевидно, $\boldsymbol{\mu}\sim\boldsymbol{\sigma} \Rightarrow \boldsymbol{\mu}\cdot\langle u\rangle\cdot\boldsymbol{\lambda}\sim\boldsymbol{\sigma}\cdot\langle u\rangle\cdot\boldsymbol{\lambda}$.

По Лемме 56, добавляемые ϕ-трассы генерируют только те ***F***-трассы, которые генерировались ϕ-трассами исходной финальной ϕ-модели. Отсюда следует, что после добавления ϕ-трасс ϕ-модель осталась конформной. Но это означает, что добавляемые ϕ-трассы на самом деле уже были в финальной ϕ-модели. В частности, была ϕ-трасса $\boldsymbol{\sigma}\cdot\langle u\rangle$. А тогда $\boldsymbol{\mu}\cdot\langle u\rangle\sim\boldsymbol{\sigma}\cdot\langle u\rangle$ & $\boldsymbol{\sigma}\cdot\langle u\rangle\in\boldsymbol{Uniform}(\boldsymbol{\Sigma})$, что противоречит нашему допущению.

3.2.2. Теперь рассмотрим случай продолжения ϕ-символом a.

3.2.2.1. Если ϕ-трасса $\boldsymbol{\mu}$ завершается ϕ-символом a, то, очевидно, ϕ-трасса $\boldsymbol{\sigma}$ завершается однородным ϕ-символом b таким, что $b_r=a_r$, и продолжение $\boldsymbol{\sigma}\cdot\langle b\rangle$ однородно. Но тогда, по ***r***-замкнутости ϕ-модели, $\boldsymbol{\mu}\sim\boldsymbol{\sigma} \Rightarrow \boldsymbol{\mu}\cdot\langle a\rangle\sim\boldsymbol{\sigma}\cdot\langle b\rangle$, что противоречит нашему допущению.

3.2.2.2. Пусть теперь ϕ-трасса $\boldsymbol{\mu}$ не завершается ϕ-символом.

Рассмотрим ϕ-символ b такой, что $b_r=a_r$, а ***gamma***-множество b_g это все действия, опасные после $\boldsymbol{\mu}\cdot\langle a\rangle$. Добавим в финальную ϕ-модель множество всех ϕ-трасс вида $\boldsymbol{\sigma}\cdot\langle b\rangle$, $\boldsymbol{\sigma}\cdot\langle b,z,\gamma\rangle$ и $\boldsymbol{\sigma}\cdot\langle b,u\rangle\cdot\boldsymbol{\lambda}$,

где $z\in b_g$, $u\in L\setminus(b_r\cup b_g)$ и $\boldsymbol{\mu}\cdot\langle a,u\rangle\cdot\boldsymbol{\lambda}\in \boldsymbol{Final}(\boldsymbol{\Sigma})$, после чего сделаем ***dr***-замыкание[38].

Легко проверяется, что при этом будут сохранены все свойства ϕ-модели, включая предельность. Например, это наглядно видно на канонической LTS-модели $\boldsymbol{\Phi 2L}\circ\boldsymbol{d}\circ\boldsymbol{Final}(\boldsymbol{\Sigma})$. Здесь добавлению ϕ-трасс с последующим ***dr***-замыканием соответствует:

1) добавление состояния $\boldsymbol{\sigma}\cdot\langle b\rangle$ и τ-перехода $\boldsymbol{\sigma}\text{—}\tau\rightarrow\boldsymbol{\sigma}\cdot\langle b\rangle$;

2) для каждого действия $z\in b_g$ добавление состояния $\boldsymbol{\sigma}\cdot\langle b,z\rangle$ и переходов $\boldsymbol{\sigma}\cdot\langle b\rangle\text{—}z\rightarrow\boldsymbol{\sigma}\cdot\langle b,z\rangle\text{—}\gamma\rightarrow\boldsymbol{\sigma}\cdot\langle b,z\rangle$;

3) для каждого действия $u\in L\setminus(b_r\cup b_g)$ добавление перехода $\boldsymbol{\sigma}\cdot\langle b\rangle\text{—}u\rightarrow\boldsymbol{\mu}\cdot\langle a,u\rangle$.

Очевидно, $\boldsymbol{\mu}\sim\boldsymbol{\sigma} \Rightarrow \boldsymbol{\mu}\cdot\langle a\rangle\sim\boldsymbol{\sigma}\cdot\langle b\rangle \;\&\; \boldsymbol{\mu}\cdot\langle a,u\rangle\cdot\boldsymbol{\lambda}\sim\boldsymbol{\sigma}\cdot\langle b,u\rangle\cdot\boldsymbol{\lambda}$.

По Лемме 56, добавляемые ϕ-трассы вида $\boldsymbol{\sigma}\cdot\langle b\rangle$ и $\boldsymbol{\sigma}\cdot\langle b,u\rangle\cdot\boldsymbol{\lambda}$ генерируют только те ***F***-трассы, которые генерировались ϕ-трассами исходной финальной ϕ-модели. Отсюда следует, что после добавления таких ϕ-трасс ϕ-модель осталась конформной. Исключением может быть только добавление ϕ-трассы вида $\boldsymbol{\sigma}\cdot\langle b,z,\gamma\rangle$ (или ϕ-трассы, получаемой из неё с помощью ***dr***-операций). Однако, поскольку $\boldsymbol{\mu}\cdot\langle a\rangle\sim\boldsymbol{\sigma}\cdot\langle b\rangle$ и действие z опасно после $\boldsymbol{\mu}\cdot\langle a\rangle$ и $\boldsymbol{\sigma}\cdot\langle b\rangle$, такое добавление также не может нарушить конформность, поскольку

---

[38] Это также эквивалентно добавлению ***dr***-замыкания указанного добавляемого множества ϕ-трасс. Также, вместо полного ***r***-замыкания достаточно вместе с каждой добавляемой ϕ-трассой вида $\boldsymbol{\sigma}\cdot\langle b\rangle\cdot\ldots$ добавлять ϕ-трассы с большим числом ϕ-символов $\boldsymbol{\sigma}\cdot\langle b,\ldots,b\rangle\cdot\ldots$, в том числе и с бесконечным числом $\boldsymbol{\sigma}\cdot\langle b,b,\ldots\rangle$.

добавляются продолжения $\mathfrak{R}$-трасс опасным после них действием и далее разрушением. Но это означает, что добавляемые ϕ-трассы на самом деле уже были в финальной ϕ-модели. В частности, была ϕ-трасса $\boldsymbol{\sigma}\cdot\langle \mathtt{b}\rangle$. А тогда $\boldsymbol{\mu}\cdot\langle \mathtt{a}\rangle\sim\boldsymbol{\sigma}\cdot\langle \mathtt{b}\rangle$ & $\boldsymbol{\sigma}\cdot\langle \mathtt{b}\rangle\in \boldsymbol{Uniform}(\boldsymbol{\Sigma})$, что противоречит нашему допущению.

Мы показали, что во всех случаях утверждение 3 верно.

95. Доказательство Леммы 59

Нам надо показать: $\boldsymbol{d}\circ\boldsymbol{Uniform}(\boldsymbol{\Sigma})\in \boldsymbol{\Phi MODEL}(\mathtt{L})$, $\boldsymbol{d}\circ\boldsymbol{Final}(\boldsymbol{\Sigma})\preccurlyeq\boldsymbol{d}\circ\boldsymbol{Uniform}(\boldsymbol{\Sigma})$ и $\boldsymbol{d}\circ\boldsymbol{Uniform}(\boldsymbol{\Sigma})$ предельна.

1. По Лемме 58, для каждой финальной ϕ-трассы найдётся такая однородная ϕ-трасса $\boldsymbol{\sigma}$, что $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$.

Отсюда $\boldsymbol{d}\circ\boldsymbol{Final}(\boldsymbol{\Sigma})\preccurlyeq\boldsymbol{d}\circ\boldsymbol{Uniform}(\boldsymbol{\Sigma})$.

2. Нам осталось показать, что ***d***-замыкание множества однородных ϕ-трасс является предельной ϕ-моделью.

Нагляднее всего это показывается на канонической LTS-модели $\boldsymbol{\Phi 2L}\circ\boldsymbol{d}\circ\boldsymbol{Final}(\boldsymbol{\Sigma})$. Нам достаточно для каждой ϕ-трассы $\boldsymbol{\mu}\cdot\langle \mathtt{a}\rangle$, где ϕ-трасса $\boldsymbol{\mu}\in\boldsymbol{Uniform}(\boldsymbol{\Sigma})$, а ϕ-символ $\mathtt{a}$ не однороден после $\boldsymbol{\mu}$, удалить τ-переход $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle \mathtt{a}\rangle$. По Лемме 58, состояние $\boldsymbol{\mu}$ остаётся нестабильным, поскольку найдётся похожая однородная ϕ-трасса $\boldsymbol{\sigma}\cdot\langle \mathtt{b}\rangle\sim\boldsymbol{\mu}\cdot\langle \mathtt{a}\rangle$ & $\boldsymbol{\sigma}\cdot\langle \mathtt{b}\rangle\in\boldsymbol{Uniform}(\boldsymbol{\Sigma})$. Поскольку ϕ-трасса $\boldsymbol{\mu}\in\boldsymbol{Uniform}(\boldsymbol{\Sigma})$, по Лемме 57, $\boldsymbol{\sigma}=\boldsymbol{\mu}$, то есть остаётся τ-переход $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle \mathtt{b}\rangle$. Тем самым, ϕ-трассы полученной после удаления τ-переходов LTS – это подмножество ϕ-трасс исходной LTS, то есть подмножество финальной ϕ-модели. Также понятно, что остаются те и

только те ϕ-трассы, которые однородны или получаются из однородных ϕ-трасс ***d***-операцией.

Предельность следует из предельности исходной наибольшей финальной ϕ-модели: предел бесконечно возрастающей цепочки однородных ϕ-трасс является финальной ϕ-трассой, которая однородна, поскольку однороден каждый её конечный префикс.

96. **Доказательство Теоремы 37**

По Лемме 59, наибольшая однородная ϕ-модель является предельной ϕ-моделью, мажорирующей наибольшую финальную ϕ-модель, и вложена в неё по определению.

По Теореме 36, наибольшая финальная ϕ-модель монотонная.

Следовательно, по Теореме 35, наибольшая однородная ϕ-модель монотонная.

97. **Доказательство Леммы 60**

Очевидно, что ***d***-замыкание сохраняет свойства префикс-замкнутости, допустимости, согласованности, ***r***-замкнутости, полноты и предельности множества ϕ-трасс, а результат ***d***-замыкания ***d***-замкнут. Поэтому нам достаточно показать, что множество сингулярных ϕ-трасс обладает указанными сохраняемыми при ***d***-замыкании свойствами. Заметим, что все эти свойства верны для множества однородных ϕ-трасс, подмножеством которого является множество сингулярных ϕ-трасс.

Сингулярная ϕ-трасса однородна, а префикс однородной ϕ-трассы однороден. Поскольку в нём все ϕ-символы сингулярны, этот префикс также сингулярен. Поэтому множество сингулярных ϕ-трасс является деревом.

Допустимость и согласованность сингулярных ϕ-трасс следует из того, что они являются однородными ϕ-трасами, которые допустимы и согласованы (Теорема 37).

Повторение сингулярного ϕ-символа, очевидно, не нарушает сингулярности ϕ-трассы. Поэтому множество сингулярных ϕ-трасс ***r***-замкнуто.

Также ***a***-операция (продолжение ϕ-трассы, заканчивающейся на ϕ-символ $\circ$ внешним действием $z \notin \circ_r$ и далее разрушением, если $z \in \circ_g$), очевидно, не нарушает сингулярности ϕ-трассы. Поэтому множество сингулярных ϕ-трасс полно.

Сингулярная ϕ-трасса однородна, а множество однородных ϕ-трасс предельно (Теорема 37). Поэтому предел бесконечно возрастающей цепочки сингулярных ϕ-трасс, является однородной ϕ-трассой. Поскольку каждый её ϕ-символ входит в некоторый сингулярный префикс, он сингулярен. Тем самым, предел сингулярен. Поэтому множество сингулярных ϕ-трасс предельно.

98. **Доказательство Леммы 61**

Для $\boldsymbol{\lambda}=\epsilon$ утверждение очевидно.

Пусть $\boldsymbol{\lambda}\neq\epsilon$.

Обозначим наибольшую монотонную ϕ-модель $\mathbf{M}=\boldsymbol{Conf}(\boldsymbol{\Sigma})$ и рассмотрим каноническую наибольшую монотонную LTS-модель $\boldsymbol{\Phi 2L}(\mathbf{M})$.

Добавим в эту LTS-модель переход $\boldsymbol{\mu} \text{—} z \rightarrow \boldsymbol{\mu}` \cdot \langle z \rangle$, если $\boldsymbol{\lambda}$ начинается с внешнего действия $z$, или переход $\boldsymbol{\mu} \text{—} \tau \rightarrow \boldsymbol{\mu}` \cdot \langle \circ \rangle$, если $\boldsymbol{\lambda}$ начинается с ϕ-символа $\circ$.

Множество ϕ-трасс полученной LTS обозначим $\mathbf{M}`$.

Мажорирование $\boldsymbol{\mu} \preccurlyeq \boldsymbol{\mu}`$ влечёт мажорирование $\boldsymbol{\mu} \cdot \boldsymbol{\lambda}` \preccurlyeq \boldsymbol{\mu}` \cdot \boldsymbol{\lambda}`$ для каждой ϕ-трассы $\boldsymbol{\lambda}`$, начинающейся с того же символа, что и $\boldsymbol{\lambda}$.

По генеративности ϕ-трасс, $\xi(\boldsymbol{\mu} \cdot \boldsymbol{\lambda}`) \preccurlyeq \xi(\boldsymbol{\mu}` \cdot \boldsymbol{\lambda}`)$.

Поэтому полученная ϕ-модель **M`** мажорируется ϕ-моделью **M** и, следовательно, ей конформна.

А ϕ-модель **M**, как множество конформных ϕ-трасс, конформна спецификации **Σ**.

Следовательно, полученная ϕ-модель **M`** является конформной реализацией.

А тогда все добавленные ϕ-трассы, в том числе **μ·λ**, уже были в ϕ-модели **M**, то есть конформны.

99. Доказательство Леммы 62

По Лемме 60, множество $d_\circ Sing(\Sigma)$ обладает всеми свойствами предельной ϕ-модели, кроме, быть может, конвергентности. Поскольку множество $d_\circ Sing(\Sigma)$ есть подмножество наибольшей монотонной ϕ-модели, оно конформно. По условию Леммы, множество $d_\circ Sing(\Sigma)$ также мажорантно. Следовательно, для того, чтобы это множество было монотонной ϕ-моделью, достаточно доказать, что оно конвергентно.

Допустим, оно не конвергентно, то есть в нём существует безопасная ϕ-трасса **μ**, не продолжаемая дивергенцией и не продолжаемая сингулярным ϕ-символом.

Но тогда в наибольшей монотонной ϕ-модели ϕ-трасса **μ** продолжается некоторым ϕ-символом ○.

По мажорантности множества $d_\circ Sing(\Sigma)$, в нём должна найтись мажоранта $\mu`\cdot\langle\circ`\rangle \succcurlyeq \mu\cdot\langle\circ\rangle$.

Но тогда, по Лемме 61, ϕ-трасса **μ** также продолжается сингулярным ϕ-символом ○` в наибольшей монотонной ϕ-модели, следовательно, $\mu\cdot\langle\circ`\rangle \in d_\circ Sing(\Sigma)$, что противоречит допущению о неконвергентности ϕ-трассы **μ**.

Итак, мы пришли к противоречию и, следовательно, множество $d_{\circ}Sing(\Sigma)$ конвергентно.

100. Доказательство Леммы 63

Преобразование Np можно выполнять как преобразование канонической LTS-модели $s=\Phi 2L(T)$ и взятие ϕ-трасс полученной LTS. Это преобразование LTS-модели заключается в том, что для состояния $\mu\cdot\langle o\rangle$ с несингулярным ϕ-символом $\phi(\mu\cdot\langle o\rangle)=o$ выполняем следующие действия:

1) Добавим новые состояния $\mu\cdot\langle o_z\rangle$, где z пробегает множество несингулярных действий ϕ-символа o.
2) Удалим τ-переход $\mu\text{—}\tau\rightarrow\mu\cdot\langle o\rangle$ и добавим τ-переход $\mu\text{—}\tau\rightarrow\mu\cdot\langle o_z\rangle$ для каждого несингулярного действия z ϕ-символа o.
3) Для каждого перехода $\mu\cdot\langle o\rangle\text{—}u\rightarrow\mu\cdot\langle o,u\rangle$, где u сингулярное действие ϕ-символа o, добавим переходы $\mu\cdot\langle o_z\rangle\text{—}u\rightarrow\mu\cdot\langle o,u\rangle$ для каждого несингулярного действия z ϕ-символа o.
4) Для каждого перехода $\mu\cdot\langle o\rangle\text{—}z\rightarrow\mu\cdot\langle o,z\rangle$, где z несингулярное действие ϕ-символа o, добавим один переход $\mu\cdot\langle o_z\rangle\text{—}z\rightarrow\mu\cdot\langle o,z\rangle$.

Преобразованную LTS обозначим $s`$.

По построению, каждое добавленное состояние $\mu\cdot\langle o_z\rangle$ имеет ϕ-символ $\phi(\mu\cdot\langle o_z\rangle)=(o_r\cup\{z\},o_g)=o_z$.

Поскольку каноническая LTS s ϕ-детерминирована (Лемма 36), а ϕ-символ o несингулярен, удалена каждая нормальная ϕ-трасса вида $\mu\cdot\langle o\rangle\cdot\lambda$, и вместо неё добавлена нормальная ϕ-трасса вида $\mu\cdot\langle o_z\rangle\cdot\lambda$ (для каждого несингулярного

действия z, с которого не начинается $\boldsymbol{\lambda}$). И наоборот: каждая добавленная нормальная ϕ-трасса имеет вид $\boldsymbol{\mu}\cdot\langle\circ_z\rangle\cdot\boldsymbol{\lambda}$, где ϕ-трасса $\boldsymbol{\mu}\cdot\langle\circ\rangle\cdot\boldsymbol{\lambda}$ была удалена.

Тем самым, $\boldsymbol{\Phi}^{\omega}(\boldsymbol{s}\grave{}) = \boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$.

Очевидно, что $\circ \preccurlyeq \circ_z$ и $\boldsymbol{\mu}\cdot\langle\circ\rangle\cdot\boldsymbol{\lambda} \preccurlyeq \boldsymbol{\mu}\cdot\langle\circ_z\rangle\cdot\boldsymbol{\lambda}$.

Тем самым, $\boldsymbol{T} = \boldsymbol{\Phi}^{\omega}(\boldsymbol{s}) \preccurlyeq \boldsymbol{\Phi}^{\omega}(\boldsymbol{s}\grave{}) = \boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$.

Поскольку ϕ-символы $\circ$ и $\circ_z$ порождают одно и то же множество $\mathfrak{R}$-отказов, каждая $\mathfrak{R}$-трасса, генерируемая удалённой нормальной ϕ-трассой $\boldsymbol{\mu}\cdot\langle\circ\rangle\cdot\boldsymbol{\lambda}$, генерируется некоторой добавленной нормальной ϕ-трассой $\boldsymbol{\mu}\cdot\langle\circ_z\rangle\cdot\boldsymbol{\lambda}$. И наоборот: каждая $\mathfrak{R}$-трасса, генерируемая добавленной нормальной ϕ-трассой $\boldsymbol{\mu}\cdot\langle\circ_z\rangle\cdot\boldsymbol{\lambda}$, генерируется удалённой нормальной ϕ-трассой $\boldsymbol{\mu}\cdot\langle\circ\rangle\cdot\boldsymbol{\lambda}$.

Тем самым, наше преобразование LTS не меняет множества её $\mathfrak{R}$-трасс.

Следовательно, $\boldsymbol{s} \sim_{\mathfrak{R}} \boldsymbol{s}\grave{}$, что влечёт $\boldsymbol{T} \sim_{\mathfrak{R}} \boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$.

101. Доказательство Теоремы 38

Будем вести доказательство от противного. Пусть есть наименьшая мажорантная конформная ϕ-модель $\boldsymbol{T}$, в которой ϕ-трасса $\boldsymbol{\mu}$ продолжается несингулярным ϕ-символом $\circ$. Без ограничения общности можно считать, что ϕ-трасса $\boldsymbol{\mu}$ не завершается ϕ-символом.

Рассмотрим преобразование $\boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$. По Лемме 63, множество $\boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$ также является ϕ-моделью, $\boldsymbol{T} \sim_{\mathfrak{R}} \boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$ и $\boldsymbol{T} \preccurlyeq \boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$.

Поскольку $\boldsymbol{T}$ конформна, $\boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$ также конформна.

Поскольку $\boldsymbol{T}$ мажорантна, $\boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu},\circ)$ также мажорантна.

Но тогда мы имеем $\mu\cdot\langle o\rangle\in T\setminus Np(T,\mu,o)$.

Поэтому $T\not\subseteq Np(T,\mu,o)$.

Но это противоречит тому, что T наименьшая мажорантная конформная ϕ-модель.

Мы пришли к противоречию и, следовательно, наше допущение не верно, а утверждение Леммы верно.

102. Доказательство Теоремы 39

По Лемме 60, множество $d\circ Sing(\Sigma)$ обладает всеми свойствами предельной ϕ-модели, кроме, быть может, конвергентности. Поскольку d-замыкание, очевидно, сохраняет конвергентность, нам достаточно показать, что при наличии наименьшей мажорантной конформной ϕ-модели множество $d\circ Sing(\Sigma)$ конвергентно. Это значит, что каждая сингулярная ϕ-трасса, не содержащая и не продолжающаяся разрушением и дивергенцией, конвергентна, то есть продолжается некоторым ϕ-символом.

Будем вести доказательство от противного. Рассмотрим каноническую LTS-модель S для наибольшей однородной ϕ-модели. Пусть в ней после сингулярной ϕ-трассы μ нет состояния с сингулярным ϕ-символом. Без ограничения общности мы можем считать, что μ нормальная ϕ-трасса, не завершающаяся ϕ-символом. Поскольку каноническая LTS ϕ-детерминирована, нормальный ϕ-маршрут с трассой μ заканчивается в единственном нестабильном состоянии μ.

В наименьшей мажорантной конформной ϕ-модели должна найтись мажоранта $\mu`\succcurlyeq\mu$.

По Теореме 38, наименьшая мажорантная конформная ϕ-модель сингулярна.

Поэтому в ней после $\mu`$ найдётся сингулярный ϕ-символ $o`$.

Выполним следующее преобразование LTS $\mathbf{s}$: добавим состояние $\boldsymbol{\mu}\cdot\langle\circ`\rangle$, проведём τ-переход $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle\circ`\rangle$ и для каждой ϕ-трассы $\boldsymbol{\mu`}\cdot\langle\circ`,z\rangle$, которая была в наименьшей мажорантной ϕ-модели, добавим переход $\boldsymbol{\mu}\cdot\langle\circ`\rangle\text{—}z\rightarrow\boldsymbol{\mu`}\cdot\langle\circ`,z\rangle$. Заметим, что состояние $\boldsymbol{\mu`}\cdot\langle\circ`,z\rangle$ должно быть в LTS $\mathbf{s}$, поскольку ϕ-трасса $\boldsymbol{\mu`}\cdot\langle\circ`,z\rangle$ нормальна и сингулярна (следовательно, однородна), LTS $\mathbf{s}$ содержит все однородные ϕ-трассы, и в ней нормальная ϕ-трасса $\boldsymbol{\mu`}\cdot\langle\circ`,z\rangle$ заканчивается в состоянии $\boldsymbol{\mu`}\cdot\langle\circ`,z\rangle$.

Все добавленные в LTS $\mathbf{s}$ ϕ-трассы имеют вид $\boldsymbol{\mu}\cdot\langle\circ`\rangle\cdot\boldsymbol{\lambda}$, где $\boldsymbol{\mu`}\cdot\langle\circ`\rangle\cdot\boldsymbol{\lambda}$ принадлежит наименьшей мажорантной конформной ϕ-модели.

Поскольку, ϕ-трассы $\boldsymbol{\mu}$ и $\boldsymbol{\mu`}\cdot\langle\circ`\rangle\cdot\boldsymbol{\lambda}$ конформны, и $\boldsymbol{\mu`}\succcurlyeq\boldsymbol{\mu}$, по Лемме 61, ϕ-трасса $\boldsymbol{\mu}\cdot\langle\circ`\rangle\cdot\boldsymbol{\lambda}$ конформна.

Поскольку, ϕ-трассы $\boldsymbol{\mu}$ и $\boldsymbol{\mu`}\cdot\langle\circ`\rangle\cdot\boldsymbol{\lambda}$ сингулярны, ϕ-трасса $\boldsymbol{\mu}\cdot\langle\circ`\rangle\cdot\boldsymbol{\lambda}$ сингулярна.

Следовательно, все добавленные в LTS $\mathbf{s}$ ϕ-трассы уже были в ней.

Но тогда $\boldsymbol{\mu}$ продолжается сингулярным ϕ-символом $\circ`$, что противоречит допущению.

103. Доказательство Леммы 64

Нам достаточно доказать, что $\boldsymbol{d}\circ\boldsymbol{Uniform}\circ\boldsymbol{F}(\mathbf{s})\preccurlyeq\boldsymbol{d}\circ\boldsymbol{Sing}\circ\boldsymbol{F}(\mathbf{s})$.

Пусть ϕ-трасса $\boldsymbol{\mu}$ произвольная однородная ϕ-трасса. Нам надо доказать, что существует её сингулярная мажоранта $\boldsymbol{\sigma}\succcurlyeq\boldsymbol{\mu}$.

Рассмотрим все однородные ϕ-трассы $\boldsymbol{\mu}$` той же длины, что ϕ-трасса $\boldsymbol{\mu}$, которые отличаются от ϕ-трасса $\boldsymbol{\mu}$ только ϕ-символами: $|\boldsymbol{\mu}`|=|\boldsymbol{\mu}|$ & $\forall i\in[1..|\boldsymbol{\mu}|]$ $\boldsymbol{\mu}`(i)=\boldsymbol{\mu}(i)\in L_{\Delta\gamma} \vee \boldsymbol{\mu}`(i)\in\Phi(L)$ & $\boldsymbol{\mu}(i)\in\Phi(L)$.

В наибольшей однородной ϕ-модели все мажоранты ϕ-трассы $\boldsymbol{\mu}$ являются α-мажорантами и находятся среди этих ϕ-трасс $\boldsymbol{\mu}$`. Одной из таких мажорант, конечно, является сама ϕ-трасса $\boldsymbol{\mu}$.

Очевидно, для КК-ветвящейся спецификации $\boldsymbol{s}$ число таких ϕ-трасс $\boldsymbol{\mu}$` конечно.

Обозначим через n суммарное число ϕ-символов, которыми продолжается одна из ϕ-трасс $\boldsymbol{\mu}$` или её префикс.

Применим n раз преобразование $\boldsymbol{Np}(\boldsymbol{T},\boldsymbol{\mu}`,o)$ к наибольшей однородной ϕ-модели $\boldsymbol{T}=\boldsymbol{d}\circ\boldsymbol{Uniform}\circ\boldsymbol{F}(\boldsymbol{s})$, каждый раз выбирая одну из её ϕ-трасс $\boldsymbol{\mu}$` или её префикс и один из продолжающих несингулярных ϕ-символов o, если, конечно, такие несингулярные ϕ-символы ещё остаются в ϕ-трассах $\boldsymbol{\mu}$`.

По Лемме 63, результат каждого применения преобразования $\boldsymbol{Np}$ является монотонной ϕ-моделью, мажорирующей операнд.

Более того, преобразование $\boldsymbol{Np}$, очевидно, сохраняет однородность.

Поскольку каждое применение преобразования $\boldsymbol{Np}$ удаляет некоторый несингулярный ϕ-символ, через n шагов у нас не останется ни одного несингулярного ϕ-символа в оставшихся ϕ-трассах $\boldsymbol{\mu}$`. Иными словами, все оставшиеся ϕ-трассы $\boldsymbol{\mu}$` не только однородны, но и сингулярны.

Но среди таких оставшихся ϕ-трасс $\boldsymbol{\mu}$` должна остаться мажоранта ϕ-трассы $\boldsymbol{\mu}$.

Следовательно, есть сингулярная мажоранта $\boldsymbol{\sigma}\succcurlyeq\boldsymbol{\mu}$.

104. **Доказательство Теоремы 40**

По Лемме 64, если спецификация КК-ветвящаяся, то ***d***-замыкание множества её сингулярных ϕ-трасс мажорантно.

По Лемме 62, если ***d***-замыкание множества сингулярных ϕ-трасс мажорантно, то оно является монотонной ϕ-моделью.

Следовательно, если спецификация КК-ветвящаяся, то ***d***-замыкание множества её сингулярных ϕ-трасс является монотонной ϕ-моделью.

105. **Доказательство Теоремы 41**

По Теореме 18, ϕ-спецификация и наибольшая однородная ϕ-модели имеют одинаковые множества безопасных $\mathfrak{R}$-трасс.

Необходимость. Нам достаточно показать, что, если в канонической наибольшей однородной LTS $\mathbf{s}$ состояние s Б-достижимо, но не БЛК-ветвящееся или не тау-ограниченное, то найдётся безопасная ϕ-трасса, продолжаемая бесконечным числом ϕ-символов.

В канонической LTS $\mathbf{s}$ в каждом Б-достижимом состоянии все τ-переходы, кроме τ-петли, заканчиваются в стабильных состояниях. Поэтому состояние s тау-ограниченное.

В канонической LTS $\mathbf{s}$ в каждом Б-достижимом состоянии определено не более одного перехода по каждому внешнему действию. Поэтому, если состояние не БЛК-ветвящееся, то в нём должно быть определено бесконечное число τ-переходов.

Если в состоянии s определена τ-петля, то любая ϕ-трасса, заканчивающаяся в этом состоянии, продолжается дивергенцией. А тогда после этой ϕ-трассы опасны все внешние действия и, следовательно, ***gamma***-множество любого однородного ϕ-символа после этой ϕ-трассы содержит все внешние действия, а ***ref***-множество (не пересекающееся с ***gamma***-множеством), тем самым, пусто.

Иными словами, такая ϕ-трасса может продолжаться только одним однородным ϕ-символом $(\varnothing, L)$. Следовательно, в состоянии s определено конечное число τ-переходов.

Поэтому остаётся один случай: в состоянии s нет τ-петли, но определено бесконечное множество τ-переходов $s \text{—}\tau\rightarrow s`$. Поскольку состояние s Б-достижимо, существует безопасная $\mathfrak{R}$-трасса μ, заканчивающаяся в состоянии s. Рассмотрим любой маршрут с этой $\mathfrak{R}$-трассой, заканчивающийся в состоянии s. Очевидно, ϕ-трасса $\boldsymbol{\mu}$ этого маршрута безопасна (генерирует безопасную $\mathfrak{R}$-трассу μ) и также заканчивается в состоянии s. Также эта ϕ-трасса $\boldsymbol{\mu}$ заканчивается во всех постсостояниях $s`$ τ-переходов, то есть в бесконечном множестве состояний. В канонической LTS $\mathbf{s}$ все эти состояния имеют разные ϕ-символы. Следовательно, ϕ-трасса $\boldsymbol{\mu}$ продолжается бесконечным числом ϕ-символов, что и требовалось доказать.

Достаточность.

По Лемме 13, в Б.БЛК+Б.О-ветвящейся канонической наибольшей однородной LTS $\mathbf{s}$ каждая безопасная $\mathfrak{R}$-трасса заканчивается в конечном множестве состояний. Каждая безопасная ϕ-трасса $\boldsymbol{\mu}$ генерирует некоторую безопасную $\mathfrak{R}$-трассу, которая заканчивается в каждом состоянии, в котором заканчивается ϕ-трасса $\boldsymbol{\mu}$ (хотя не обязательно только в этих состояниях). Следовательно, ϕ-трасса $\boldsymbol{\mu}$ заканчивается в конечном числе состояний $s`$. Далее ϕ-трасса $\boldsymbol{\mu}$ продолжается ϕ-символом $o$ тогда и только тогда, когда одно из этих состояний $s`$ стабильно и имеет ϕ-символ o. Следовательно, ϕ-трасса $\boldsymbol{\mu}$ продолжается конечным числом ϕ-символов.

106. **Доказательство Леммы 65**

1. Необходимость. Если $\mathfrak{R}$-трасса $\mu\cdot\langle\Delta\rangle$ конформна, то все $\mathfrak{R}$-кнопки опасны после μ. При наличии пустой $\mathfrak{R}$-кнопки, $\varnothing\in\mathfrak{R}$, это возможно для безопасной $\mathfrak{R}$-трассы μ только тогда, когда $\mu\cdot\langle\Delta\rangle\in\boldsymbol{F}(\mathbf{S})$. При отсутствии пустой $\mathfrak{R}$-кнопки, $\varnothing\notin\mathfrak{R}$, это возможно только тогда, когда для безопасной $\mathfrak{R}$-трассы μ все действия опасны после μ, а тогда $\boldsymbol{gamma}(\mu)=\mathtt{L}$. Таким образом, должно быть выполнено условие $\boldsymbol{true} = \mu\cdot\langle\Delta\rangle\in\boldsymbol{F}(\mathbf{S}) \vee \boldsymbol{gamma}(\mu)=\mathtt{L} \;\&\; \varnothing\notin\mathfrak{R} = \boldsymbol{div}(\mu)$.

2. Достаточность. Если $\mu\cdot\langle\Delta\rangle\in\boldsymbol{F}(\mathbf{S})$, то $\mathfrak{R}$-трасса $\mu\cdot\langle\Delta\rangle$ конформна как любая $\mathfrak{R}$-трасса спецификации (конформность рефлексивна). Пусть нет пустой $\mathfrak{R}$-кнопки, $\varnothing\notin\mathfrak{R}$, и $\boldsymbol{gamma}(\mu)=\mathtt{L}$. Добавим в $\boldsymbol{F}(\mathbf{S})$ множество $\mathfrak{R}$-трасс $\boldsymbol{di}(\mu\cdot\langle\Delta\rangle)$. Покажем, что получится $\mathfrak{R}$-модель. Нагляднее всего это видно на LTS-модели $\boldsymbol{F2L}\circ\boldsymbol{F}(\mathbf{S})$. Здесь добавлению этого множества $\mathfrak{R}$-трасс соответствует добавление τ-петли в состоянии μ. Покажем, что полученная $\mathfrak{R}$-модель конформна. Действительно, конформность может нарушиться только в том случае, когда $\mathfrak{R}$-трасса μ безопасна, и некоторая $\mathfrak{R}$-кнопка $\mathtt{P}$ безопасна после μ. Поскольку пустой $\mathfrak{R}$-кнопки нет, существует действие $\mathtt{z}\in\mathtt{P}$, которое, тем самым, безопасно после μ. Но тогда $\mathtt{z}\notin\boldsymbol{gamma}(\mu)$, чего быть не может, поскольку $\boldsymbol{gamma}(\mu)=\mathtt{L}$.

107. Доказательство Леммы 66

1. Покажем, что в исходной LTS-спецификации **S** после тау-трассы μ существует такое состояние s, которое порождает каждую (μ,o)-безопасную трассу ρ: $\rho \in \xi_{\circ}\phi(s)$.

Допустим, это не так.

По Лемме 13, множество состояний **S** ***after*** μ конечно; обозначим эти состояния $s_1,\ldots,s_n$.

Для каждого состояния s_i можно выбрать (μ,o)-безопасную трассу ρ_i, которая не порождается состоянием s_i.

Возьмём конкатенацию этих трасс $\rho=\rho_1\cdot\ldots\cdot\rho_n$.

Очевидно, что трасса ρ (μ,o)-безопасна.

Тогда она должна порождаться одним из состояний $s_1,\ldots,s_n$, чего быть не может, так как каждое состояние s_i не порождает ρ_i, следовательно не может порождать и ρ.

2. Возьмём множество U всех таких состояний из **S** ***after*** μ, которые порождают все (μ,o)-безопасные трассы $\rho_1,\rho_2,\ldots$. Докажем, что есть такая (μ,o)-безопасная трасса ρ, которая порождается только состояниями из множества U.

Допустим, это не так.

Выберем произвольную (μ,o)-безопасную трассу ρ_1. Для неё найдётся состояние $s_1 \notin U$, порождающее ρ_1. Тогда это состояние не порождает некоторую (μ,o)-безопасную трассу ρ_2.

Возьмём конкатенацию $\rho_1\cdot\rho_2$, которая, очевидно, (μ,o)-безопасна. Для неё найдётся состояние $s_2 \notin U$, порождающее $\rho_1\cdot\rho_2$. Состояние s_1 не порождает ρ_2, а состояние s_2 порождает $\rho_1\cdot\rho_2$ и, следовательно, порождает ρ_2. Поэтому $s_2 \notin \{s_1\}$. Состояние s_2 не порождает некоторую (μ,o)-безопасную трассу ρ_3.

Возьмём конкатенацию $\rho_1 \cdot \rho_2 \cdot \rho_3$, которая, очевидно, (μ, o)-безопасна. Для неё найдётся состояние $s_3 \notin U$, порождающее $\rho_1 \cdot \rho_2 \cdot \rho_3$. Состояние s_1 не порождает ρ_2, состояние s_2 не порождает ρ_3, а состояние s_3 порождает $\rho_1 \cdot \rho_2 \cdot \rho_3$ и, следовательно, порождает ρ_2 и ρ_3. Поэтому $s_3 \notin \{s_1, s_2\}$. Состояние s_3 не порождает некоторую (μ, o)-безопасную трассу ρ_4.

И так далее. У нас получится бесконечная цепочка различных состояний $s_1, s_2, s_3 \ldots$, что противоречит конечности множества состояний **S** ***after*** μ.

Итак, мы доказали, что существует такая (μ, o)-безопасная трасса ρ, которая порождается теми и только теми состояниями, которые порождают каждую (μ, o)-безопасную трассу $\rho`$. Следовательно, **S** ***after*** $\mu \cdot \rho = U \subseteq$ **S** ***after*** $\mu \cdot \rho`$.

108. Доказательство Леммы 67

Будем вести доказательство индукцией по ϕ-трассе **μ**.

1. Пустая безопасная ϕ-трасса генерирует единственную безопасную $\mathfrak{R}$-трассу ϵ. Очевидно, $\zeta(\epsilon) = \{\epsilon\}$, **S** ***after*** $\epsilon \subseteq$ **S** ***after*** ϵ.

2. Пусть утверждение верно для ϕ-трассы **μ**, и её главной трассы $\mu \in \zeta(\boldsymbol{\mu})$. Рассмотрим возможные безопасные (оставляющие ϕ-трассу безопасной) продолжения ϕ-трассы **μ**.

2.1. Продолжение внешним действием z.

Рассмотрим трассу $\mu \cdot \langle z \rangle$. Безопасные $\mathfrak{R}$-трассы, генерируемые ϕ-трассой $\boldsymbol{\mu} \cdot \langle z \rangle$, это все $\mathfrak{R}$-трассы вида $\mu` \cdot \langle z \rangle$, где $\mu`$ безопасная $\mathfrak{R}$-трасса, генерируемая ϕ-трассой **μ**, а действие z безопасно после $\mu`$. Далее **S** ***after*** $\mu \subseteq$ **S** ***after*** $\mu`$ влечёт **S** ***after*** $\mu \cdot \langle z \rangle \subseteq$ **S** ***after*** $\mu` \cdot \langle z \rangle$.

Значит действие z безопасно после μ. А тогда $\mathfrak{R}$-трасса $\mu\cdot\langle z\rangle$ безопасна и генерируется ϕ-трассой $\boldsymbol{\mu}\cdot\langle z\rangle$. Значит $\mu\cdot\langle z\rangle\in\zeta(\boldsymbol{\mu}\cdot\langle z\rangle)$.

2.2. Продолжение ϕ-символом o.

По Лемме 66, существует главная трасса $\rho=\zeta(\mu,o)$. Рассмотрим трассу $\mu\cdot\rho$. Безопасные $\mathfrak{R}$-трассы, генерируемые ϕ-трассой $\boldsymbol{\mu}\cdot\langle o\rangle$, это все $\mathfrak{R}$-трассы вида $\mu`\cdot\rho`$, где $\mu`$ безопасная $\mathfrak{R}$-трасса, генерируемая $\phi$-трассой $\boldsymbol{\mu}$, а $\rho`$ $(\mu`,o)$-безопасная трасса $\mathfrak{R}$-отказов. Поэтому $\mathfrak{R}$-трасса $\mu\cdot\rho$ безопасна и генерируется $\phi$-трассой $\boldsymbol{\mu}\cdot\langle o\rangle$. Поскольку $\mathbf{S}\ \textit{after}\ \mu\subseteq\mathbf{S}\ \textit{after}\ \mu`$, трасса $\rho`$ $(\mu,o)$-безопасна. А тогда $\mathbf{S}\ \textit{after}\ \mu\cdot\rho\subseteq\mathbf{S}\ \textit{after}\ \mu\cdot\rho`$, следовательно, $\mathbf{S}\ \textit{after}\ \mu\cdot\rho\subseteq\mathbf{S}\ \textit{after}\ \mu`\cdot\rho`$. Значит $\mu\cdot\rho\in\zeta(\boldsymbol{\mu}\cdot\langle o\rangle)$.

109. Доказательство Леммы 68

По Лемме 67, главная трасса существует.

Если действие z безопасно после главной трассы μ, то, поскольку главная трасса безопасна, действие z безопасно после ϕ-трассы $\boldsymbol{\mu}$.

Если действие z безопасно после ϕ-трассы $\boldsymbol{\mu}$, то найдётся такая безопасная $\mathfrak{R}$-трасса $\mu`$, генерируемая $\phi$-трассой $\boldsymbol{\mu}$, после которой действие $z$ безопасно. Далее $\mathbf{S}\ \textit{after}\ \mu\subseteq\mathbf{S}\ \textit{after}\ \mu`$ влечёт $\mathbf{S}\ \textit{after}\ \mu\cdot\langle z\rangle\subseteq\mathbf{S}\ \textit{after}\ \mu`\cdot\langle z\rangle$. Значит действие z безопасно после главной трассы μ.

110. Доказательство Леммы 69

Если внешнее действие $z \in o_g$, то оно опасно после ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$. Следовательно, по Лемме 68, оно опасно после её главной трассы μ. Следовательно, $z \in \boldsymbol{gamma}(\mu)$.

Если внешнее действие $z \in \boldsymbol{gamma}(\mu)$, то оно опасно после μ. Тогда, по Лемме 68, оно опасно после $\boldsymbol{\mu}\cdot\langle o\rangle$. Поскольку ϕ-символ o однороден, должно быть $z \in o_g$.

111. Доказательство Леммы 70

Пусть $\mu \in \zeta(\boldsymbol{\mu}\cdot\langle o\rangle)$.

1. Покажем, что $\boldsymbol{ref}(\mu) \subseteq o_r$.

По Лемме 69, $\boldsymbol{gamma}(\mu) = o_g$. Нам достаточно показать, что для любого внешнего действия $z \in L \setminus (o_r \cup o_g)$ имеет место $z \in L \setminus \boldsymbol{ref}(\mu)$.

В силу однородности ϕ-символа o, такое действие z безопасно после $\boldsymbol{\mu}\cdot\langle o\rangle$ и продолжает $\boldsymbol{\mu}\cdot\langle o\rangle$.

Следовательно, действие z безопасно после главной трассы μ и продолжает её (продолжение $\mu\cdot\langle z\rangle$ безопасно в наибольшей однородной модели и, следовательно, безопасно в исходной спецификации).

Тем самым, $z \in L \setminus \boldsymbol{ref}(\mu)$.

2. Покажем, что $\mathfrak{R}\circ\boldsymbol{ref}(\mu) = \mathfrak{R}(o_r)$.

Допустим, это не так.

Поскольку мы уже доказали, что $\boldsymbol{ref}(\mu) \subseteq o_r$, должно быть $\mathfrak{R}\circ\boldsymbol{ref}(\mu) \subset \mathfrak{R}(o_r)$. Пусть некоторый $\mathfrak{R}$-отказ $P \subseteq o_r$, но $P \not\subseteq \boldsymbol{ref}(\mu)$.

Покажем, что $\mathfrak{R}$-трасса $\mu\cdot\langle P\rangle$ безопасна.

Действительно, если бы это было не так, нашлось бы внешнее действие $z\in P$ такое, что $\mu\cdot\langle z,\gamma\rangle\in \boldsymbol{F}(\mathbf{s})$.

Следовательно, действие z опасно после μ, то есть $z\in \boldsymbol{gamma}(\mu)$.

А тогда, по Лемме 69, действие $z\in o_g$.

Однако $z\in P$ и $P\subseteq o_r$ влечёт $z\in o_r$, и, значит, множества o_r и o_g имеют непустое пересечение, чего быть не может.

Итак, $\mathfrak{R}$-трасса $\mu\cdot\langle P\rangle$ безопасна и, очевидно, генерируется ϕ-трассой $\boldsymbol{\mu}\cdot\langle o\rangle$.

Следовательно, $\mathbf{s}\ \boldsymbol{after}\ \mu \subseteq \mathbf{s}\ \boldsymbol{after}\ \mu\cdot\langle P\rangle$.

Однако $P\not\subseteq \boldsymbol{ref}(\mu)$ влечёт наличие такого действия $z\in P$, что $\mu\cdot\langle z\rangle\in \boldsymbol{F}(\mathbf{s})$.

Тогда в некотором состоянии из $\mathbf{s}\ \boldsymbol{after}\ \mu$ есть переход по z и, следовательно, нет отказа P.

Но это противоречит $\mathbf{s}\ \boldsymbol{after}\ \mu \subseteq \mathbf{s}\ \boldsymbol{after}\ \mu\cdot\langle P\rangle$.

112. Доказательство Леммы 71

Рассмотрим наибольшую однородную LTS $\mathbf{s}` = \boldsymbol{F2L}\circ\boldsymbol{d}\circ\boldsymbol{Uniform}\circ\boldsymbol{F}(\mathbf{s})$.

В ней ϕ-трассе $\boldsymbol{\mu}\cdot\langle a\rangle$ соответствует состояние $\boldsymbol{\mu}\cdot\langle a\rangle$.

Выполним следующее преобразование LTS:

1) добавим состояние $\boldsymbol{\mu}\cdot\langle b\rangle$;

2) добавим переход $\boldsymbol{\mu}\text{—}\tau\rightarrow\boldsymbol{\mu}\cdot\langle b\rangle$;

3) для каждого действия $z\in L\setminus b_r$ добавим переход $\boldsymbol{\mu}\cdot\langle b\rangle\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle a,z\rangle$.

Поскольку $a\trianglelefteq b$, имеем $a_g=b_g$ & $a_r\subseteq b_r$ & $\mathfrak{R}(a_r)=\mathfrak{R}(b_r)$.

Поскольку $a_r\subseteq b_r$, п.3) преобразования определён корректно: состояние $\boldsymbol{\mu}\cdot\langle a,z\rangle$ существует.

По Лемме 43, все добавленные ϕ-трассы имеют вид $\boldsymbol{\mu}\cdot\langle b\rangle\cdot\boldsymbol{\lambda}$, где ϕ-трасса $\boldsymbol{\mu}\cdot\langle a\rangle\cdot\boldsymbol{\lambda}$ была раньше.

Поскольку $\mathfrak{R}(a_r)=\mathfrak{R}(b_r)$, мы не добавили ни одной новой $\mathfrak{R}$-трассы.

Следовательно, LTS $\mathbf{s}`$ конформна.

Из вида добавленных ϕ-трасс $\boldsymbol{\mu}\cdot\langle b\rangle\cdot\boldsymbol{\lambda}$ следует, что они финальны и однородны, если были финальны и однородны ϕ-трассы $\boldsymbol{\mu}\cdot\langle a\rangle\cdot\boldsymbol{\lambda}$.

Следовательно, LTS $\mathbf{s}`$ однородна.

А тогда ϕ-трасса $\boldsymbol{\mu}\cdot\langle b\rangle$ однородна.

113. **Доказательство Леммы 72**

1. Покажем, что power-состояние вида $[\mu]$ или γ нестабильно. Трасса $\mu\in \boldsymbol{Tau}(\mathbf{s})$. В силу конвергентности $\mathfrak{R}$-модели, трасса μ полна или продолжается некоторым $\mathfrak{R}$-отказом. Тем самым для некоторой (быть может, пустой) трассы $\mathfrak{R}$-отказов ρ $\mathfrak{R}$-трасса $\mu\cdot\rho$ стабильна. А тогда, по правилу вывода (3), в LTS $\boldsymbol{P}(\mathbf{s})$ существует τ-переход $[\mu]\text{—}\tau\rightarrow([\mu\cdot\rho],\boldsymbol{ref}(\mu\cdot\rho))$. Следовательно, power-состояние $[\mu]$ нестабильно. Состояние γ нестабильно, поскольку, по правилу вывода (1), в нём определён γ-переход.

2. Покажем, что power-состояние вида $([\mu],r)$ стабильно.

По правилам вывода, τ- определяются только в power-состоянии вида $[\mu]$, а γ-переходы – только в состоянии γ. Тем самым, в power-состоянии вида $([\mu],r)$ не определены τ- и γ-переходы, то есть оно стабильно.

114. **Доказательство Леммы 73**

Очевидно, что по условиям Леммы начальное power-состояние $p_0=[\epsilon]$.

Будем вести доказательство индукцией по маршруту R.

База индукции. Пустой маршрут нормален, поскольку, по Лемме 72, начальное power-состояние нестабильно, и заканчивается в начальном power-состоянии. Пустой маршрут имеет пустую нормальную ϕ-трассу, генерирующую только пустую $\mathfrak{R}$-трассу. Если $p_0=[\epsilon]$, то $\langle\gamma\rangle\notin \boldsymbol{F}(\mathbf{S})$, поэтому пустая $\mathfrak{R}$-трасса безопасна. Тем самым для $R=\epsilon$, $\boldsymbol{\mu}=\epsilon$, $\mu=\epsilon$ и любой $\mathfrak{R}$-трассы $\sigma\in\xi(\boldsymbol{\mu})$ имеем:

$\omega(R)=[\mu]$, $\mu\in\xi(\boldsymbol{\mu})$, $\mu\in \boldsymbol{Safe}_\circ\boldsymbol{F}(\mathbf{S})$, $\sigma\in\boldsymbol{F}(\mathbf{S})$ и $[\mu]\subseteq[\sigma]$.

Шаг индукции. Предположим, что утверждение доказано для маршрута R, имеющего нормальную ϕ-трассу $\boldsymbol{\mu}$ и заканчивающегося в power-состоянии $[\mu]$ или $([\mu],r)$, и докажем его для маршрута $R`$, получающегося из R с помощью ещё одного перехода. Мы рассмотрим три возможных случая.

A. Переход по безопасному внешнему действию z.

По правилам вывода, такой переход ведёт в power-состояние $[\mu\cdot\langle z\rangle]$.

По Лемме 72, power-состояние $[\mu\cdot\langle z\rangle]$ нестабильно.

Поскольку маршрут R имеет нормальную ϕ-трассу $\boldsymbol{\mu}$, маршрут $R`$ имеет нормальную ϕ-трассу $\boldsymbol{\mu}\cdot\langle z\rangle$.

Поскольку $\mu\in\xi(\boldsymbol{\mu})$, имеем $\mu\cdot\langle z\rangle\in\xi(\boldsymbol{\mu})\cdot\{\langle z\rangle\}=\xi(\boldsymbol{\mu}\cdot\langle z\rangle)$.

Поскольку $\mu\in \boldsymbol{Safe}_\circ\boldsymbol{F}(\mathbf{S})$ и, по правилам вывода, $z\notin \boldsymbol{gamma}(\mu)$, имеем z ***safe*** $\boldsymbol{F}(\mathbf{S})$ ***after*** μ и $\mu\cdot\langle z\rangle\in \boldsymbol{Safe}_\circ\boldsymbol{F}(\mathbf{S})$.

Любая $\mathfrak{R}$-трасса из $\xi(\boldsymbol{\mu}\cdot\langle z\rangle)$ имеет вид $\sigma\cdot\langle z\rangle$, где $\sigma\in\xi(\boldsymbol{\mu})$.

Поскольку $\sigma \in \boldsymbol{F}(\mathbf{s})$ и $[\mu] \subseteq [\sigma]$, имеем $[\mu \cdot \langle z \rangle] \subseteq [\sigma \cdot \langle z \rangle]$.

Поскольку $[\mu \cdot \langle z \rangle] \neq \varnothing$, имеем $[\sigma \cdot \langle z \rangle] \neq \varnothing$, то есть $\sigma \cdot \langle z \rangle \in \boldsymbol{F}(\mathbf{s})$.

Итак $\omega(R`) = [\mu \cdot \langle z \rangle]$, $\mu \cdot \langle z \rangle \in \xi(\boldsymbol{\mu} \cdot \langle z \rangle)$, $\mu \cdot \langle z \rangle \in \boldsymbol{Safe} \circ \boldsymbol{F}(\mathbf{s})$, $\sigma \cdot \langle z \rangle \in \boldsymbol{F}(\mathbf{s})$ и $[\mu \cdot \langle z \rangle] \subseteq [\sigma \cdot \langle z \rangle]$.

B. τ-переход вида $[\mu] \text{---} \tau \rightarrow ([\mu], r)$.

Поскольку маршрут R имеет нормальную ϕ-трассу $\boldsymbol{\mu}$, маршрут $R`$ имеет нормальную $\phi$-трассу $\boldsymbol{\mu}` = \boldsymbol{\mu} \cdot \langle \phi([\mu], r) \rangle$.

Поскольку $\mu \in \xi(\boldsymbol{\mu})$, имеем $\mu \in \xi(\boldsymbol{\mu}`)$.

Также имеем $\mu \in \boldsymbol{Safe} \circ \boldsymbol{F}(\mathbf{s})$.

Переход $[\mu] \text{---} \tau \rightarrow ([\mu], r)$ существует, если μ полная трасса.

Тогда, по правилам вывода, в power-состояние $([\mu], r)$ определён переход хотя бы по одному действию из каждого $\mathfrak{R}$-отказа, то есть это power-состояние не порождает $\mathfrak{R}$-отказов.

Поэтому любая $\mathfrak{R}$-трасса $\sigma \in \xi(\boldsymbol{\mu}`)$ также $\sigma \in \xi(\boldsymbol{\mu})$.

Также имеем $[\mu] \subseteq [\sigma]$ и $\sigma \in \boldsymbol{F}(\mathbf{s})$.

Итак: $\omega(R`) = [\mu]$, $\mu \in \xi(\boldsymbol{\mu}`)$, $\mu \in \boldsymbol{Safe} \circ \boldsymbol{F}(\mathbf{s})$, $\sigma \in \boldsymbol{F}(\mathbf{s})$, $[\mu] \subseteq [\sigma]$.

C. τ-переход вида $[\mu] \text{---} \tau \rightarrow ([\mu \cdot \rho], r)$, где ρ непустая трасса $\mathfrak{R}$-отказов.

Поскольку маршрут R имеет нормальную ϕ-трассу $\boldsymbol{\mu}$, маршрут $R`$ имеет нормальную $\phi$-трассу $\boldsymbol{\mu}` = \boldsymbol{\mu} \cdot \langle \phi([\mu \cdot \rho], r) \rangle$.

По Лемме 72, power-состояние $([\mu \cdot \rho], r)$ стабильно.

$\mathfrak{R}$-трассы из $\xi(\boldsymbol{\mu}`)$ – это все трассы вида $\sigma \cdot \rho`$, где $\sigma \in \xi(\boldsymbol{\mu})$ и $\rho`$ трасса $\mathfrak{R}$-отказов, порождаемых power-состоянием $([\mu \cdot \rho], r)$.

Поскольку $\mu\in\xi(\boldsymbol{\mu})$, а ρ порождается power-состоянием $([\mu\cdot\rho],r)$, $\mu\cdot\rho\in\xi(\boldsymbol{\mu}`)$.

По правилам вывода, $\mu\cdot\rho\in\boldsymbol{Stab}(\mathbf{S})$, следовательно, $\mu\cdot\rho\in\boldsymbol{Safe}\circ\boldsymbol{F}(\mathbf{S})$.

Очевидно, что каждое состояние $s\in[\mu\cdot\rho]$ стабильно.

По правилам вывода, $\mathfrak{R}$-отказ порождается power-состоянием $([\mu\cdot\rho],r)$ тогда и только тогда, когда он порождается каждым состоянием $s\in[\mu\cdot\rho]$.

Следовательно, трасса $\mathfrak{R}$-отказов $\rho`$ порождается power-состоянием $([\mu\cdot\rho],r)$ тогда и только тогда, когда она порождается каждым состоянием $s\in[\mu\cdot\rho]$.

Поскольку $[\mu]\subseteq[\sigma]$ и $\sigma\in\boldsymbol{F}(\mathbf{S})$, имеем $[\mu\cdot\rho]\subseteq[\mu\cdot\rho`]\subseteq[\sigma\cdot\rho`]$.

Поскольку $[\mu\cdot\rho]\neq\varnothing$, имеем $[\sigma\cdot\rho`]\neq\varnothing$, то есть $\sigma\cdot\rho`\in\boldsymbol{F}(\mathbf{S})$.

Итак: $\omega(R`)=([\mu\cdot\rho],r)$, $\mu\cdot\rho\in\xi(\boldsymbol{\mu}`)$, $\mu\cdot\rho\in\boldsymbol{Safe}\circ\boldsymbol{F}(\mathbf{S})$, $\sigma\in\boldsymbol{F}(\mathbf{S})$, $[\mu]\subseteq[\sigma]$.

115. Доказательство Леммы 74

По Лемме 73, каждая $\mathfrak{R}$-трасса σ power-LTS, порождаемая нормальной ϕ-трассой $\boldsymbol{\mu}$ конечного маршрута R, заканчивающегося в power-состоянии $[\mu]$ или $([\mu],r)$, принадлежит спецификации $\mathbf{S}$. Следовательно, $\sigma\preccurlyeq_{\mathfrak{R}}\boldsymbol{F}(\mathbf{S})$.

Другие $\mathfrak{R}$-трассы имеют один из следующих видов:

1) $\sigma\cdot\langle\Delta\rangle$, где $\sigma\in\xi(\boldsymbol{\mu})$, если маршрут R заканчивается в power-состоянии $[\mu]$, в котором определена τ-петля согласно правилу вывода (2).
2) $\sigma\cdot\langle z\rangle$ или $\sigma\cdot\langle z,\gamma\rangle$, где $\sigma\in\xi(\boldsymbol{\mu})$, если маршрут R заканчивается в power-состоянии $[\mu]$ или $([\mu],r)$, в котором определён разрушающий переход по действию z согласно правилу вывода (6) или (7).

Случай 1) возможен тогда, когда $\boldsymbol{div}(\mu)=\boldsymbol{true}$.

По Лемме 65, $\mathfrak{R}$-трасса $\mu\cdot\langle\Delta\rangle$ конформна.

А тогда все $\mathfrak{R}$-кнопки опасны в спецификации $\mathbf{S}$ после μ.

По Лемме 73, для каждой $\mathfrak{R}$-трассы $\sigma\in\xi(\boldsymbol{\mu})$ имеет место $\sigma\in\boldsymbol{F}(\mathbf{S})$ & $[\mu]\subseteq[\sigma]$, следовательно, $\mathfrak{R}$-кнопки опасны в спецификации $\mathbf{S}$ после σ.

А тогда $\boldsymbol{div}(\sigma)=\boldsymbol{true}$, и, по Лемме 65, $\mathfrak{R}$-трасса $\sigma\cdot\langle\Delta\rangle$ конформна.

Тогда, по Теореме 24, $\sigma\cdot\langle\Delta\rangle\preccurlyeq_{\mathfrak{R}}\boldsymbol{F}(\mathbf{S})$.

Случай 2) возможен тогда, когда $z\in\boldsymbol{gamma}(\mu)$, то есть z ***safe*** $\mathbf{S}$ ***after*** μ.

По Лемме 73, для каждой $\mathfrak{R}$-трассы $\sigma\in\xi(\boldsymbol{\mu})$ имеет место $\sigma\in\boldsymbol{F}(\mathbf{S})$ & $[\mu]\subseteq[\sigma]$, следовательно, z ***safe*** $\mathbf{S}$ ***after*** σ.

А тогда $\sigma\cdot\langle z\rangle\preccurlyeq_{\mathfrak{R}}\boldsymbol{F}(\mathbf{S})$ и $\sigma\cdot\langle z,\gamma\rangle\preccurlyeq_{\mathfrak{R}}\boldsymbol{F}(\mathbf{S})$.

Итак, мы получили $\boldsymbol{F}\circ\boldsymbol{P}(\mathbf{S})\preccurlyeq_{\mathfrak{R}}\boldsymbol{F}(\mathbf{S})$.

Тогда по Теореме 24, $\boldsymbol{P}(\mathbf{S})$ ***saco*** $\mathbf{S}$.

116. Доказательство Леммы 75

По Лемме 74, power-LTS конформна. Нам надо показать, что она финальная и однородна.

1. Сначала покажем, что power-LTS финальна.

По Лемме 73, каждая нормальная ϕ-трасса $\boldsymbol{\mu}$ конечного маршрута R, заканчивающегося в power-состоянии $[\mu]$ или $([\mu],r)$, безопасна и, следовательно, финальна.

Любая другая нормальная ϕ-трасса – это продолжение $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ или $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$, а также префикс последней ϕ-трассы $\boldsymbol{\mu}\cdot\langle z\rangle$. Поскольку все эти продолжения конформны, они финальны.

Любая ϕ-трасса получается из нормальной ϕ-трассы с помощью ***r***-операций, оставляющих ϕ-трассу финальной, и затем ***d***-операций.

Поэтому все ϕ-трассы power-LTS – это ***d***-замыкание её финальных ϕ-трасс, то есть power-LTS финальна.

2. Теперь покажем, что power-LTS однородна.

Покажем, что нормальная ϕ-трасса $\boldsymbol{\mu}$ конечного маршрута R, заканчивающегося в power-состоянии $[\mu]$, продолжается только однородными ϕ-символами.

Действие z, опасное после ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$, – это действие, опасное после каждой безопасной $\mathfrak{R}$-трассы $\sigma\in\xi(\boldsymbol{\mu}\cdot\langle o\rangle)$.

По правилам вывода, найдётся такая трасса $\mathfrak{R}$-отказов ρ, что $o=\phi((\,[\mu\cdot\rho],\boldsymbol{ref}(\mu\cdot\rho)))$.

По Лемме 73, $[\mu\cdot\rho]\subseteq[\sigma]$.

Поэтому действие z опасно после каждой безопасной $\mathfrak{R}$-трассы $\sigma\in\xi(\boldsymbol{\mu}\cdot\langle o\rangle)$ тогда и только тогда, когда оно опасно после $\mu\cdot\rho$.

Поэтому нам нужно доказать, что $o_g=\boldsymbol{gamma}(\mu\cdot\rho)$.

А это непосредственно следует из правила вывода (7).

Любая другая нормальная ϕ-трасса – это продолжение $\boldsymbol{\mu}\cdot\langle\Delta\rangle$ или $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$, а также префикс последней ϕ-трассы $\boldsymbol{\mu}\cdot\langle z\rangle$. Поскольку все эти продолжения не добавляют новых ϕ-символов, они однородны.

Любая ϕ-трасса получается из нормальной ϕ-трассы с помощью ***r***-операций, оставляющих ϕ-трассу однородной, и затем ***d***-операций.

Поэтому все ϕ-трассы power-LTS – это замыкание её однородных ϕ-трасс, то есть power-LTS однородна.

117. Доказательство Леммы 76

1. Сначала покажем, что каждая тау-трасса $\mu \in \mathit{Tau}(\mathbf{S})$ имеется в power-LTS $\boldsymbol{P}(\mathbf{S})$ и заканчивается в power-состоянии $[\mu]$.

По Лемме 73, если power-состояние $[\mu]$ достижимо, то в нём заканчивается тау-трасса μ.

Поэтому нам достаточно показать, что каждое power-состояние $[\mu]$, где $\mu \in \mathit{Tau}(\mathbf{S})$, достижимо.

Будем вести доказательство индукцией по тау-трассе μ.

База индукции. Из существования power-состояния $[\epsilon]$ следует, что это начальное power-состояние и $\epsilon \in \mathit{Tau}(\mathbf{S})$.

Шаг индукции. Допустим утверждение верно для тау-трассы $\mu \in \mathit{Tau}(\mathbf{S})$ и докажем его для тау-трасс 1) $\mu \cdot \langle z \rangle \in \mathit{Tau}(\mathbf{S})$, 2) $\mu \cdot \rho \cdot \langle z \rangle \in \mathit{Tau}(\mathbf{S})$, где $z \in L$ и $\rho \in \mathfrak{R}^*$.

Для каждого случая есть соответствующее правило вывода, которое определяет переход:

1) $[\mu] \text{---} z \rightarrow [\mu \cdot \langle z \rangle]$,

2) $[\mu] \text{---} \tau \rightarrow ([\mu \cdot \rho], \mathit{ref}(\mu \cdot \rho)) \text{---} z \rightarrow [\mu \cdot \rho \cdot \langle z \rangle]$.

Нам надо показать, что $[\mu] \in (\boldsymbol{P}(\mathbf{S})\ \mathit{after}\ \mu)$ влечёт:

1) $[\mu \cdot \langle z \rangle] \in (\boldsymbol{P}(\mathbf{S})\ \mathit{after}\ \mu \cdot \langle z \rangle)$,

2) $[\mu \cdot \rho \cdot \langle z \rangle] \in (\boldsymbol{P}(\mathbf{S})\ \mathit{after}\ \mu \cdot \rho \cdot \langle z \rangle)$.

Случай 1) очевиден. Для случая 2) достаточно показать, что в power-состоянии $([\mu \cdot \rho], \mathit{ref}(\mu \cdot \rho))$ есть трасса отказов ρ, то есть в этом power-состоянии реализуется каждый отказ $P \in \mathit{Im}(\rho)$. А это непосредственно следует из правил вывода, определяющих в таком power-состоянии переход

по действию z только в том случае, когда $\mu\cdot\rho\cdot\langle z\rangle\in \mathit{Tau}(\mathbf{s})$, то есть когда $z\notin\cup_{\circ}\mathit{Im}(\rho)$.

2. Теперь покажем, что тау-трасса $\mu\in \mathit{Tau}(\mathbf{s})$ является главной трассой некоторой ϕ-трассы.

Это непосредственно следует из Леммы 73.

118. Доказательство Леммы 77

По правилу вывода (3), имеется переход $[\mu]\text{—}\tau\rightarrow s$, где $s=([\mu\cdot\rho],\mathit{ref}(\mu\cdot\rho))$.

По Лемме 76, в power-состоянии $[\mu]$ заканчивается тау-трасса μ.

Следовательно, в power-состояния s заканчивается стабильная трасса $\mu\cdot\rho$.

Докажем минимальность ϕ-символа состояния s.

a) $\phi(s)_g=\mathit{gamma}(\mu\cdot\rho)$. Это следует из правила вывода (7).

a) $\phi(s)_r=\mathit{ref}(\mu\cdot\rho)$. Это следует из правила вывода (5).

119. Доказательство Теоремы 42

1. Сначала покажем, что в наибольшей однородной модели имеются все тау-трассы и все они продолжаются всеми своими минимальными ϕ-символами.

Действительно:

1.1. по Лемме 76, power-LTS содержит все тау-трассы;

1.2. по Лемме 77, в power-LTS каждая тау-трасса продолжается всеми своими минимальными ϕ-символами;

1.3. по Лемме 75, power-LTS однородна.

2. По **1** и Лемме 71, в наибольшей однородной модели имеются все тау-трассы и все они продолжаются всеми своими доминирующими ϕ-символами.

3. Теперь покажем, что в наибольшей однородной модели каждый ϕ-символ, которым продолжается тау-трасса, доминирует некоторый минимальный ϕ-символ этой тау-трассы.

Действительно:

3.1. по Лемме 70, в наибольшей однородной модели каждый ϕ-символ доминирует некоторый минимальный ϕ-символ, но, правда, не после любой продолжаемой им тау-трассы, а только после главной трассы ϕ-трассы;

3.2. в то же время, по Лемме 76, в наибольшей однородной модели каждая тау-трасса является главной трассой некоторой ϕ-трассы.

Итак, по **2** и **3**, в наибольшей однородной модели имеются все тау-трассы и все они продолжаются всеми своими доминирующими ϕ-символами и только ими, что и требовалось доказать.

120. **Доказательство Леммы 78**

В power-LTS любой τ-переход имеет вид $[\mu] \text{—}\tau\rightarrow [\mu]$ или $[\mu] \text{—}\tau\rightarrow ([\mu\cdot\rho], \mathit{ref}(\mu\cdot\rho))$. В первом случае τ-переход является петлёй в power-состоянии. Во втором случае, по Лемме 72, τ-переход ведёт в стабильное состояние.

Отсюда следует, что нам достаточно показать, что из каждого power-состояния power-LTS выходит конечное число переходов по каждому символу (внешнему действию, разрушению или τ).

В power-LTS, согласно правилам вывода, в каждом power-состоянии определяется не более одного перехода по каждому внешнему действию или разрушению.

Если LTS-спецификация $\mathbf{s}$ Б.БЛК+Б.О-ветвящаяся, то, по Лемме 13, каждая безопасная $\mathfrak{R}$-трасса, в частности тау-трасса μ, заканчивается в конечном множестве её состояний.

Поскольку у конечного множества конечное число подмножеств, а каждый τ-переход из power-состояния $[\mu]$ ведёт в power-состояние вида $([\mu\cdot\rho],\boldsymbol{ref}(\mu\cdot\rho))$ и $[\mu\cdot\rho]\subseteq[\mu]$, «веер» τ-переходов из power-состояния $[\mu]$ конечен.

121. Доказательство Леммы 79

Пусть ϕ-символ a имеет бесконечное множество A несингулярных действий. Рассмотрим множество ϕ-символов:

$$\boldsymbol{D}(a) = \{a_z \mid z\in A\} = \{a_z \mid z\in L\setminus a_r \;\&\; \mathfrak{R}(a_r)=\mathfrak{R}(a_r\cup\{z\}\},$$

где $a_z=(a_r\cup\{z\},a_g)$.

Рассмотрим произвольную доминанту D ϕ-символа a.

Очевидно, для любого ϕ-символа $a_z\in\boldsymbol{D}(a)$ пересечение $\boldsymbol{D}(a,z)=D\cap a_z{}^{\triangle}$ является доминантой ϕ-символа a_z.

Покажем, что для любых двух разных несингулярных действий $z_1\neq z_2$ эти пересечения разные $\boldsymbol{D}(a,z_1)\neq\boldsymbol{D}(a,z_2)$.

Действительно, поскольку $\boldsymbol{D}(a,z_1)$ доминанта для a_{z1}, для каждого $d_1\in\boldsymbol{D}(a,z_1)$ имеет место $a_r\cup\{z_1\}=a_{z1r}\subseteq d_{1r}$, следовательно $z_1\in d_{1r}$.

Поскольку $z_1\notin a_r\cup\{z_2\}=a_{z2r}$, а $\boldsymbol{D}(a,z_2)$ доминанта для a_{z2}, должно найтись $d_2\in\boldsymbol{D}(a,z_2)$ такое, что $z_1\notin d_{2r}$.

Таким образом, $d_2\in\boldsymbol{D}(a,z_2)$ и $d_2\notin\boldsymbol{D}(a,z_1)$.

Если число несингулярных действий бесконечно, то $\boldsymbol{D}(a)$ бесконечно.

Тогда, если бы доминанта D была конечна, в неё было бы вложено бесконечное множество различных множеств $\boldsymbol{D}(a,z)$, чего быть не может, так как конечное множество имеет конечное число различных подмножеств.

Значит, доминанта бесконечна, что и требовалось доказать.

122. **Доказательство Леммы 80**

1. Необходимость. Достаточно показать, что, если множество несингулярных действий ϕ-символа бесконечно, то верхний конус бесконечен. Очевидно, что верхний конус ϕ-символа является доминантой этого ϕ-символа. По Лемме 79, если число несингулярных действий ϕ-символа бесконечно, то любая доминанта этого ϕ-символа бесконечна, в частности, должен быть бесконечен верхний конус.

2. Достаточность. Пусть множество A несингулярных действий ϕ-символа a конечно. Рассмотрим произвольный доминирующий ϕ-символ $b_z \in a^{\triangle}$. Тогда $a_g=b_g$, $a_r \subseteq b_r$ и $\mathfrak{R}(a_r)=\mathfrak{R}(b_r)$. Поскольку добавление сингулярного действия к ***ref***-множеству a_r меняет множество порождаемых $\mathfrak{R}$-отказов, должно быть $b_r \setminus a_r \subseteq A$. Поскольку множество A конечно, число таких ***ref***-множеств b_r конечно. Следовательно, конечно число доминирующих ϕ-символов, то есть конечен верхний конус $a^{\triangle}$.

123. **Доказательство Леммы 81**

Пусть у ϕ-символа a существует конечная доминанта.

1. Сначала покажем, что ϕ-символ a имеет сингулярную доминанту.

По Лемме 79, число несингулярных действий ϕ-символа a конечно.

А тогда, по Лемме 80, верхний конус ϕ-символа a по доминированию конечен.

Для произвольного ϕ-символа o с множеством несингулярных действий O обозначим множество ϕ-символов, получающихся добавлением в ***ref***-множество o_r одного несингулярного действия:

$$\boldsymbol{D}(o) = \{o_z \mid z \in O\} = \{o_z \mid z \in L \setminus o_r \ \&\ \mathfrak{R}(o_r) = \mathfrak{R}(o_r \cup \{z\})\},$$

где $o_z = (o_r \cup \{z\}, o_g)$.

Для множества ϕ-символов A обозначим $\boldsymbol{E}(A) = \cup\{\boldsymbol{D}(o) \mid o \in A\}$.

Очевидно, что $\boldsymbol{E}(a^{\triangle})$ является доминантой ϕ-символа a.

Если множество $a^{\triangle}$ содержит хотя бы один несингулярный ϕ-символ, то $a^{\triangle} \supset \boldsymbol{E}(a^{\triangle})$.

Далее, если множество $\boldsymbol{E}(a^{\triangle})$ содержит хотя бы один несингулярный ϕ-символ, то $\boldsymbol{E}(a^{\triangle}) \supset \boldsymbol{E} \circ \boldsymbol{E}(a^{\triangle})$ и множество $\boldsymbol{E} \circ \boldsymbol{E}(a^{\triangle})$ также является доминантой ϕ-символа a.

Будем продолжать эту операцию до тех пор, пока остаётся хотя бы один несингулярный ϕ-символ. В результате мы получим цепочку вложенных множеств: $a^{\triangle} \supset \boldsymbol{E}(a^{\triangle}) \supset \boldsymbol{E} \circ \boldsymbol{E}(a^{\triangle}) \supset \boldsymbol{E} \circ \boldsymbol{E} \circ \boldsymbol{E}(a^{\triangle}) \supset \ldots$.

Поскольку верхний конус $a^{\triangle}$ конечен, эта цепочка должна оборваться, то есть на конечном шаге мы получим сингулярную доминанту.

2. Теперь покажем, что любой сингулярный ϕ-символ $b \in a^{\triangle}$ принадлежит любой доминанте ϕ-символа a.

Действительно, верхний конус сингулярного ϕ-символа b – это либо множество $\{b\}$, если нет несингулярных действий, либо множество $\{b, b_z\}$, где $b_z = (b_r \cup \{z\}, b_g)$, если имеется единственное несингулярное

действие z, причём ϕ-символ b_z сингулярен. При этом верхний конус является единственной предоминантой и единственной доминантой ϕ-символа b.

Поэтому любая доминанта ϕ-символа a должна содержать верхний конус любого сингулярного ϕ-символа из $a^{\triangle}$, что эквивалентно тому, что она содержит все сингулярные ϕ-символы из $a^{\triangle}$.

Таким образом, существует сингулярная доминанта, и она содержит (как всякая доминанта) все сингулярные ϕ-символы из $a^{\triangle}$. Следовательно, эта сингулярная доминанта совпадает с множеством всех сингулярных ϕ-символов из $a^{\triangle}$.

Поскольку любая доминанта содержит все сингулярные ϕ-символа из $a^{\triangle}$, она включает сингулярную доминанту. Тем самым, сингулярная доминанта является наименьшей доминантой.

124. **Доказательство Теоремы 43**

По Лемме 76, power-LTS содержит все тау-трассы;

По Лемме 77, в power-LTS каждая тау-трасса продолжается всеми своими минимальными ϕ-символами;

По Лемме 75, power-LTS однородна.

По Лемме 78, power-LTS ЛК+О-ветвящаяся.

Следовательно, в наибольшей однородной модели, подмоделью которой является power-LTS, каждая тау-трасса продолжается конечным числом минимальных ϕ-символов.

По Лемме 80, верхний конус каждого ϕ-символа конечен.

Верхний конус ϕ-символа, очевидно, является его доминантой.

Следовательно, для каждого ϕ-символа существует конечная доминанта.

А тогда, по Лемме 81, множество сингулярных ϕ-символов из верхнего конуса каждого ϕ-символа является доминантой, и эта доминанта конечна (как подмножество конечного множества).

Следовательно, множество всех сингулярных ϕ-символов, которыми продолжается тау-трасса, конечно (как объединение конечного числа конечных множеств) и доминирует все доминирующие ϕ-символы тау-трассы.

По Теореме 42, каждая тау-трасса продолжается в наибольшей однородной модели всеми своими доминирующими ϕ-символами и только такими ϕ-символами.

Следовательно, множество всех сингулярных ϕ-символов, которыми продолжается тау-трасса, конечно и доминирует все ϕ-символы, которыми продолжается тау-трасса.

Наибольшая сингулярная модель получается из power-LTS заменой каждого power-состояния s с минимальным ϕ-символом на конечное множество power-состояний $s_1, s_2, \ldots$ с доминирующими этот ϕ-символ сингулярными ϕ-символами, и проведением из каждого такого power-состояния s_i тех переходов, которые выходили из power-состояния s и которые допускаются ϕ-символом power-состояния s_i.

Тем самым, наибольшая сингулярная модель существует, является монотонной и ЛК+О-ветвящейся.

125. **Доказательство Леммы 82**

По определению мажорирования ϕ-трасс, из равенства ***gamma***-множеств и вложенности ***ref***-множеств следует мажорирование ϕ-символов. Нам надо доказать обратную импликацию.

По Лемме 55, достаточно рассмотреть только α-мажорирование ϕ-трасс.

Будем вести доказательство индукцией по длине ϕ-трасс.

Пустая ϕ-трасса α-мажорируется только сама собой, и утверждение Леммы очевидно.

Пусть утверждение верно для ϕ-трассы **μ** и докажем его для её продолжения базовым символом (действием, дивергенцией или разрушением) или ϕ-символом.

Продолжение базовым символом z. Если $\boldsymbol{\mu}\cdot\langle z\rangle\preccurlyeq\boldsymbol{\sigma}`$, то найдётся такая ϕ-трасса **σ**, что $\boldsymbol{\sigma}`=\boldsymbol{\sigma}\cdot\langle z\rangle$, где $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$. Отсюда, если утверждение Леммы верно для ϕ-трассы **μ**, то оно верно и для ϕ-трассы $\boldsymbol{\mu}\cdot\langle z\rangle$.

Продолжение ϕ-символом a. Поскольку после ϕ-символа не может следовать другой ϕ-символ, достаточно рассмотреть случай, когда ϕ-трасса **μ** не завершается ϕ-символом. Если $\boldsymbol{\mu}\cdot\langle a\rangle\preccurlyeq\boldsymbol{\sigma}`$, то найдётся такая ϕ-трасса **σ**, что $\boldsymbol{\sigma}`=\boldsymbol{\sigma}\cdot\langle b\rangle$, где $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$ и $a\preccurlyeq b$.

Сначала докажем, что $a_r\subseteq b_r$.

Допустим, это не верно. Тогда найдётся действие $z\in a_r\setminus b_r$.

Тогда, поскольку $a\preccurlyeq b$ влечёт $a_r\subseteq b_r\cup b_g$, должно быть $z\in b_g$.

Так как $a_r\cap a_g=\varnothing$, $z\notin a_g$.

Из однородности следует, что действие z безопасно после ϕ-трассы $\boldsymbol{\mu}\cdot\langle a\rangle$.

Из финальности следует, что ϕ-трасса $\boldsymbol{\mu}\cdot\langle a\rangle$ безопасна.

Рассмотрим главную трассу ϕ-трассы $\boldsymbol{\mu}\cdot\langle a\rangle$. Очевидно, она безопасна и имеет вид $\mu\cdot\rho$, где μ главная трасса ϕ-трассы **μ**, а $\rho\in\xi(a)$.

По Лемме 68, действие z безопасно после $\mathfrak{R}$-трассы $\mu\cdot\rho$.

По предположению шага индукции, утверждение Леммы верно для $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$.

Тогда, очевидно, $\xi(\boldsymbol{\mu}) \subseteq \xi(\boldsymbol{\sigma})$.

Следовательно, $\rho \in \xi(\boldsymbol{\sigma})$.

Если бы $\rho \in \xi(b)$, то, поскольку $z \in b_g$, действие z было бы опасно после $\mathfrak{R}$-трассы $\mu \cdot \rho$, что не верно. Следовательно, $\rho \notin \xi(b)$.

Поскольку $\epsilon \in \xi(b)$, найдётся такая трасса $\mathfrak{R}$-отказов $\rho_1 \in \xi(b)$ и такой $\mathfrak{R}$-отказ $P \not\subseteq b_r$, что $\rho_1 \cdot \langle P \rangle \leq \rho$.

Поскольку $a_r \subseteq b_r \cup b_g$ и $\rho \in \xi(b)$, должно быть $P \subseteq a_r \subseteq b_r \cup b_g$.

Но тогда найдётся действие $t \in P \cap b_g$.

Поскольку $\mu \in \xi(\boldsymbol{\mu})$ и $\rho_1 \in \xi(b)$, действие t опасно после $\mu \cdot \rho_1$.

А это противоречит тому, что $\rho_1 \cdot \langle P \rangle \leq \rho$ и $\mathfrak{R}$-трасса $\mu \cdot \rho$ безопасна.

Мы пришли к противоречию, и, следовательно, наше допущение не верно, и $a_r \subseteq b_r$.

Теперь покажем, что $a_g = b_g$.

Поскольку $\xi(\boldsymbol{\mu}) \subseteq \xi(\boldsymbol{\sigma})$ и $a_r \subseteq b_r$, имеем $\xi(\boldsymbol{\mu} \cdot \langle a \rangle) \subseteq \xi(\boldsymbol{\sigma} \cdot \langle b \rangle)$.

Поэтому любое действие, опасное после $\boldsymbol{\sigma} \cdot \langle b \rangle$, опасно после $\boldsymbol{\mu} \cdot \langle a \rangle$.

А тогда из однородности следует, что $a_g \supseteq b_g$.

Поскольку $a \preccurlyeq b$ влечёт $a_g \subseteq b_g$, имеем $a_g = b_g$.

126. **Доказательство Леммы 83**

Сначала покажем, что для конечного множества индексов I размером n множество B должно содержать не менее $n-1$ элементов. Действительно, пусть это не так. Будем первый индекс называть строкой, а второй индекс – столбцом. Тем самым, мы имеем матрицу $n \times n$, в клетках которой размещается менее $n-1$ различных элементов. Тогда существует элемент x, которому

соответствует не менее $n+1$ пар индексов ij, то есть $x=b(i_1j_1)=b(i_2j_2)=\ldots=b(i_{n+1}j_{n+1})$. Для пары индексов i_kj_k в столбце i_k и в строке j_k не может располагаться элемент x. Отмечаем эти строку и столбец. Выполняем это для $k=1,2,\ldots,n$. Очевидно, мы отметим все строки и столбцы матрицы. Поэтому не может быть, чтобы $x=b(i_{n+1}j_{n+1})$. Мы пришли к противоречию, и, следовательно, наше допущение не верно, и множество B должно содержать не менее $n-1$ элементов.

Далее заметим, что для бесконечного множества индексов I мы имеем бесконечную матрицу, у которой для любого n её подматрица $n\times n$ обладает тем же свойством: для каждых $i\neq j$ и $i\neq k$ выполнено $b(i,j)\neq b(k,i)$. Следовательно, в этой подматрице не менее $n-1$ элементов. Тем самым, в бесконечной матрице не менее $n-1$ элементов для каждого n. Это возможно только в том случае, когда множество B бесконечно.

127. Доказательство Теоремы 44

Пусть в наибольшей однородной модели существует ϕ-трасса $\boldsymbol{\mu}\cdot\langle a\rangle$, где a ϕ-символ с бесконечным числом несингулярных действий.

Допустим, утверждение Леммы не верно: существует монотонная Б.БЛК+Б.О-ветвящаяся однородная LTS-модель $\boldsymbol{M}$.

Обозначим несингулярные действия ϕ-символа a через z_i, где индекс i пробегает бесконечное множество индексов I.

Рассмотрим все ϕ-трассы вида $\boldsymbol{\mu}\cdot\langle a_i\rangle$, где $a_i=(a_r\cup\{z_i\},a_g)$.

Очевидно, что все такие ϕ-трассы должны присутствовать в наибольшей однородной модели.

В модели $\boldsymbol{M}$ для каждой пары индексов $i\neq j$ должна найтись мажорирующая ϕ-трасса $\boldsymbol{\mu}_{ij}\cdot\langle a(i,j)\rangle\succcurlyeq\boldsymbol{\mu}\cdot\langle a_i\rangle$, где ϕ-символ $a(i,j)\succcurlyeq a_i$, продолжаемая действием z_j.

Тогда $z_i \in a(i,j)_r \cup a(i,j)_g$ и $z_j \notin a(i,j)_r$.

По Лемме 82, $z_i \in a(i,j)_r$ и $z_j \notin a(i,j)_r$.

Для любого $k \neq i$ будет $z_i \notin a(k,i)_r$ и $z_j \in a(k,i)_r$.

Обозначая $b(i,j)=a(i,j)_r$, имеем: для любых $i \neq j$ и $i \neq k$ выполнено $b(i,j) \neq b(k,i)$.

Тем самым, выполнены условия Леммы 83, и, следовательно, семейство ***ref***-множеств $b(i,j)$ бесконечно.

Отсюда бесконечно семейство ϕ-символов $a(i,j)$ и мажорирующих ϕ-трасс $\mu_{ij} \cdot \langle a(i,j) \rangle$.

Более того, очевидно, бесконечно число мажорирующих ϕ-трасс $\mu_{ij} \cdot \langle a(i,j) \rangle$ без повторных ϕ-символов.

Заметим, что все эти ϕ-трассы имеют одну и ту же подтрассу внешних действий.

Однако, если LTS $\mathbf{M}$ Б.БЛК+Б.О-ветвящаяся, то в ней не может быть бесконечного числа ϕ-трасс без повторных ϕ-символов с одной и той же подтрассой внешних действий.

Мы пришли к противоречию, и, следовательно, наше допущение не верно, а утверждение Леммы верно.

128. Доказательство Теоремы 45

Для доказательства достаточно рассмотреть каноническую LTS-модель ***Φ2L***$(\mathbf{\Sigma})$, где $\mathbf{\Sigma}$ монотонная Б.БЛК+Б.О-ветвящаяся LTS-модель.

Пусть ϕ-трасса μ продолжается неоднородным ϕ-символом a.

Определим соответствующий однородный ϕ-символ $b=(a_r \backslash c, a_g \backslash c)$, где c множество действий опасных после ϕ-трассы $\mu \cdot \langle a \rangle$.

Заменим состояние $\boldsymbol{\mu}\cdot\langle a\rangle$ на состояние $\boldsymbol{\mu}\cdot\langle b\rangle$, и каждый переход $\boldsymbol{\mu}\cdot\langle a\rangle\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle a,z\rangle$, где $z\in c$, на переходы $\boldsymbol{\mu}\cdot\langle b\rangle\text{—}z\rightarrow\boldsymbol{\mu}\cdot\langle b,z\rangle\text{—}\gamma\rightarrow\boldsymbol{\mu}\cdot\langle b,z\rangle$.

Выполним такую замену для каждого неоднородного ϕ-символа.

Очевидно, при этом конформность, мажорантность и Б.БЛК+Б.О-ветвимость нарушены не будут, а все ϕ-символы станут однородными.

Таким образом, мы получим монотонную Б.БЛК+Б.О-ветвящуюся однородную LTS-модель.

129. Доказательство Леммы 84

По Лемме 76 и Лемме 77, в power-LTS каждое power-состояние, кроме состояния γ, Б-достижимо.

Преобразованная LTS $\mathcal{T}(\mathbf{s})$ получается из power-LTS заменой каждого power-состояния s с минимальным ϕ-символом на конечное множество power-состояний $s_1, s_2, \ldots$ с доминирующими этот ϕ-символ сингулярными ϕ-символами, и проведением из каждого такого power-состояния s_i тех переходов, которые выходили из power-состояния s и которые допускаются ϕ-символом power-состояния s_i.

Отсюда следует, что в LTS $\mathcal{T}(\mathbf{s})$ также каждое power-состояние, кроме состояния γ, Б-достижимо.

130. Доказательство Теоремы 46

По построению, Лемме 84 и Теореме 43, для конечно-доминируемой LTS-спецификации $\mathbf{s}$ преобразованная LTS $\mathcal{T}(\mathbf{s})$ является монотонной, ЛК+О-ветвящейся наибольшей сингулярной моделью.

А тогда, по Лемме 40, LTS $\mathcal{T}(\mathbf{s})$ предельна.

131. **Доказательство Теоремы 47**

1. Покажем, что LTS **C** Н.ЛК-ветвящаяся: в каждом неразрушающем состоянии (a,b) определено конечное число переходов по каждому внешнему действию z, а также конечное число τ-переходов.

Действительно, если состояние (a,b) неразрушающее, то оба состояния-операнды a и b неразрушающие. В силу Н.ЛК-ветвимости операндов, в них определено конечное число переходов по каждому внешнему действию z. Поскольку композиционный переход по действию z наследует переход по этому действию в одном из операндов, число композиционных переходов по действию z конечно.

Композиционный τ-переход может быть асинхронным или синхронным. Асинхронный τ-переход наследует τ-переход в одном из операндов, а число таких переходов конечно в силу Н.ЛК-ветвимости операндов. Синхронному τ-переходу $(a,b)\text{—}\tau\rightarrow(a`,b`)$ соответствует пара переходов в операндах по противоположным синхронным действиям $a\text{—}z\rightarrow a`$ и $b\text{—}\underline{z}\rightarrow b`$, где $z\in A\cap\underline{B}$ и $\underline{z}\in\underline{A}\cap B$. Число таких переходов в операндах конечно, поскольку $A\cap\underline{B}\subseteq A`$, $\underline{A}\cap B\subseteq B`$, LTS **A** A`-ветвящаяся и LTS **B** B`-ветвящаяся (условие 2 в Определении 133).

2. Покажем, что LTS **C** Н.О-ветвящаяся: в каждом неразрушающем состоянии (a,b) по τ-маршрутам достижимо конечное множество состояний.

Действительно, по доказанному, дерево τ-маршрутов, начинающихся в состоянии (a,b), конечно-ветвящееся. Поэтому бесконечное множество состояний могло бы быть достигнуто только по некоторому бесконечному τ-маршруту P. Если состояние (a,b) неразрушающее, то оба состояния-операнды a и b неразрушающие. А тогда, в силу Н.О-ветвимости операндов, из этих состояний по τ-маршрутам достижимо конечное

множество состояний. Следовательно, τ-маршрут P может появиться только как результат композиции двух маршрутов в LTS-операндах, каждый из которых имеет трассу в синхронном алфавите и, по крайней мере, один проходит через бесконечное число состояний. Однако этого не может быть, поскольку $A\cap\underline{B}\subseteq A`$, $\underline{A}\cap B\subseteq B`$, LTS **A** A`-ветвящаяся и LTS **B** B`-ветвящаяся (условие 3 в Определении 133).

3. Покажем, что LTS **C** П-ветвящаяся: в каждом достижимом состоянии (a,b) дерево τ-маршрутов перечислимо-ветвящееся.

Это эквивалентно тому, что во всех достижимых состояниях определено перечислимое множество τ-переходов. Такие τ-переходы делятся на асинхронные и синхронные. Асинхронные τ-переходы наследуются из операндов, где их множества перечислимы в силу П-ветвимости операндов. Синхронный τ-переход получается композицией двух переходов по противоположным синхронным действиям. Множества таких переходов в операндах перечислимы, поскольку $A\cap\underline{B}\subseteq A`$, $\underline{A}\cap B\subseteq B`$, LTS **A** A`-ветвящаяся и LTS **B** B`-ветвящаяся (условие 1 в Определении 133).

4. Покажем, что для LTS **C** выполнено условие 1 из Определения 133: в каждом достижимом состоянии определено перечислимое множество переходов суммарно по всем действиям из алфавита C`.

Действительно, каждый такой переход является асинхронным и наследует переход в одном из операндов по асинхронному действию $z\in C`=(A`\setminus\underline{B})\cup(B`\setminus\underline{A})$. Поскольку LTS **A** A`-ветвящаяся и LTS **B** B`-ветвящаяся, множества таких переходов в операндах перечислимы (условие 1 в Определении 133).

5. Покажем, что для LTS **C** выполнено условие 2 из Определения 133: в каждом достижимом состоянии определено конечное множество неразрушающих переходов суммарно по всем действиям из алфавита C`.

Действительно, каждый такой переход является асинхронным и наследует переход в одном из операндов по асинхронному действию $z \in C` = (A` \setminus \underline{B}) \cup (B` \setminus \underline{A})$. Если композиционный переход неразрушающий, значит соответствующий переход в операнде также неразрушающий. Поскольку LTS **A** $A`$-ветвящаяся и LTS **B** $B`$-ветвящаяся, множества таких переходов в операндах конечны (условие 2 в Определении 133).

6. Покажем, что для LTS **C** выполнено условие 3 из Определения 133: нет маршрута, проходящего через бесконечное число состояний, трасса которого имеет бесконечный постфикс в алфавите $C`$.

Действительно, такой маршрут P может появиться только как результат композиции двух маршрутов в LTS-операндах: левый маршрут имеет трассу с бесконечным постфиксом в алфавите $A` \cup (A \cap \underline{B})$, а правый маршрут имеет трассу с бесконечным постфиксом в алфавите $B` \cup (\underline{A} \cap B)$, и, по крайней мере, один, для определённости, левый, проходит через бесконечное число состояний. Тогда, поскольку $A \cap \underline{B} \subseteq A`$, в левом операнде имеется маршрут, проходящий через бесконечное число состояний, трасса которого имеет бесконечный постфикс в алфавите $A`$. Но этого не может быть, поскольку LTS **A** $A`$-ветвящаяся (условие 3 в Определении 133).

132. Доказательство Теоремы 48

1. Необходимость.

По Теореме 46, LTS **T** ЛК+О-ветвящаяся. Следовательно, в каждом достижимом состоянии определено конечное число τ-переходов.

По условию 2 в Определении 133, в каждом достижимом состоянии определено конечное число неразрушающих переходов.

Тем самым, под-LTS **T`**, порождённая всеми неразрушающими переходами, конечно-ветвящаяся.

Покажем, что LTS **T`** содержит конечное число достижимых состояний.

Допустим, это не так. Тогда рассмотрим дерево её конечных маршрутов без самопересечений. Оно конечно-ветвящееся. Поскольку число состояний бесконечно, а каждое достижимое состояние достижимо по маршруту без самопересечений, все эти состояния должны лежать на дереве, и, следовательно, дерево бесконечно. А в бесконечном конечно-ветвящемся дереве есть бесконечный маршрут.

Поскольку все маршруты без самопересечений, этот бесконечный маршрут должен содержать бесконечное число различных состояний, что противоречит условию 3 в Определении 133.

При конечном числе достижимых состояний и конечном ветвлении имеем суммарно конечное число переходов из всех достижимых состояний под-LTS $\mathfrak{T}`$, то есть её конечность.

2. Достаточность.

Нам нужно показать выполнение условий 2 и 3 из Определения 133, что непосредственно следует из конечности под-LTS $\mathfrak{T}`$.

133. Доказательство Теоремы 49

1. Необходимость.

Если в исходной LTS нарушено условие 1 из Определения 135, то некоторая безопасная $\mathfrak{R}$-трасса μ продолжается неперечислимым множеством действий из алфавита $L`$. Тогда в преобразованной LTS в соответствующем достижимом power-состоянии $[\mu]$ или $([\mu], s_r)$ будет определено неперечислимое множество переходов по этим действиям, то есть будет нарушено условие 1 из Определения 133.

Если в исходной LTS нарушено условие 2 из Определения 135, то некоторая безопасная $\mathfrak{R}$-трасса μ продолжается бесконечным множеством безопасных действий из алфавита $L`$. Тогда в преобразованной LTS в

соответствующем достижимом power-состоянии $[\mu]$ или $([\mu], s_r)$ будет определено бесконечное множество неразрушающих переходов по этим действиям, то есть будет нарушено условие 2 из Определения 133.

Если в исходной LTS нарушено условие 3 из Определения 135, то некоторая бесконечная безопасная $\mathfrak{R}$-трасса μ проходит через бесконечное множество состояний и имеет бесконечный постфикс в алфавите $L`$. Тогда в преобразованной LTS имеется бесконечный маршрут, трасса которого совпадает с подтрассой базовых символов $\mathfrak{R}$-трассы $\mu$ и, следовательно, имеет тот же бесконечный постфикс в алфавите $L`$. Этот маршрут проходит через бесконечное число power-состояний, поскольку им соответствуют конечные множества состояний после конечных префиксов $\mathfrak{R}$-трассы μ, а суммарно эти множества состояний должны покрывать бесконечное множество состояний, через которые проходит $\mathfrak{R}$-трасса μ. Тем самым, будет нарушено условие 3 из Определения 133.

2. Достаточность.

Если в преобразованной LTS нарушено условие 1 из Определения 133, то в некотором достижимом power-состоянии $[\mu]$ или $([\mu], s_r)$ будет определено неперечислимое множество переходов по действиям из алфавита $L`$. Но тогда, поскольку по каждому из этих действий в преобразованной LTS имеется только один переход, в исходной LTS $\mathfrak{R}$-трасса $\mu$ продолжается неперечислимым множеством действий из алфавита $L`$, то есть будет нарушено условие 1 из Определения 135.

Если в преобразованной LTS нарушено условие 2 из Определения 133, то в некотором достижимом power-состоянии $[\mu]$ или $([\mu], s_r)$ будет определено бесконечное множество неразрушающих переходов по действиям из алфавита $L`$. Но тогда, поскольку по каждому из этих действий в преобразованной LTS имеется только один переход, в исходной LTS $\mathfrak{R}$-

трасса μ продолжается бесконечным множеством безопасных действий из алфавита $L`$, то есть будет нарушено условие 2 из Определения 135.

Если в преобразованной LTS нарушено условие 3 из Определения 133, то в ней имеется бесконечный маршрут, проходящий через бесконечное множество power-состояний и имеющий трассу с бесконечным постфиксом из алфавита $L`$. Но тогда в исходной LTS имеется бесконечно возрастающая цепочка безопасных $\mathfrak{R}$-трасс, предел которой имеет тот же постфикс. Поскольку исходная LTS Б.БЛК+Б.О-ветвящаяся, этот предел является бесконечной безопасной $\mathfrak{R}$-трассой μ. Поскольку бесконечному множеству power-состояний преобразованной LTS соответствует бесконечное множество состояний исходной LTS, $\mathfrak{R}$-трасса μ проходит через бесконечное число состояний. Тем самым, будет нарушено условие 3 из Определения 135.