САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

Парийская Екатерина Юрьевна

СИМВОЛЬНАЯ ВЕРИФИКАЦИЯ НЕПРЕРЫВНО–ДИСКРЕТНЫХ СИСТЕМ. АЛГОРИТМ ОБОБЩЕННОГО ТАЙМЕР–ПРЕОБРАЗОВАНИЯ

05.13.18 — теоретические основы математического моделирования, численные методы и комплексы программ

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научные руководители:
доктор технических наук,
профессор Карпов Ю.Г.
кандидат ф.-м. наук,
доцент Сениченков Ю.Б.

Санкт-Петербург 2000

Содержание

Введение

Теория динамических систем является традиционной и в то же время очень современной областью математических исследований. Динамические системы имеют многочисленные приложения в различных областях естественных наук, так как на их основе можно моделировать разнообразные управляемые системы и сложные динамические процессы любой природы: физической, химической, биологической, технической, производственной, экономической. Описание реальных объектов допускает большое разнообразие – от систем дифференциальных уравнений до функций алгебры логики, графов, марковских цепей и т.д. Выбор способа описания зависит от целей исследования и определяет возможные подходы к исследованию. Современные теории управления и оптимального управления, теория автоматов, алгебраическая теория линейных систем фактически являются различными разделами теории динамических систем. В середине 60-х годов они были впервые объединены в математическую теорию систем.

Сложность математических моделей управляемых систем постоянно увеличивается вследствие научно–технического прогресса. Возникают новые проблемы, связанные с параллельностью, распределенностью, иерархией в сложных системах, постоянно повышаются требования к эффективности, безопасности и надежности систем автоматического управления. Интенсивное использование электронно–вычислительной техники в системах автоматического управления на современном этапе компьютеризации общества привело к необходимости проектирования и анализа динамических систем, объединяющих в себе непрерывные и дискретные компоненты, и выделения их в отдельный класс "непрерывно–дискретных" систем.

Типичными приложениями непрерывно-дискретных систем могут служить средства комплексной автоматизации, создаваемые на базе автоматики, телемеханики, электроники и вычислительной техники, крупные производственные, энергетические, гидротехнические комплексы с автоматизированным управлением, комплексы вычислительных машин, предназначенные для обработки и планирования, экономические и транспортные системы, системы слежения, надеж-

ные системы управления физическими процессами с аварийными ситуациями, системы, состоящие из параллельных взаимодействующих динамических подсистем.

В качестве конкретного примера можно привести систему космического слежения, состоящую из нескольких параллельно работающих (и при необходимости заменяющих друг друга) локаторов, каждый из которых является динамической подсистемой. В такой системе управления появление объекта в области видимости некоторого локатора является событием, определяющим его динамику на период слежения. Основным требованием к такой системе управления, очевидно, является надежность работы всей системы при возможных отказах отдельных подсистем и максимальное количество одновременно ведомых объектов. Произвольность траектории ведомого объекта и момента его появления, возможность отказов отдельных элементов делает систему событийно-управляемой и усложняет процесс ее моделирования и анализа. Аналогичным примером может служить система управления наблюдениями на оптических и радиотелескопах сложной структуры, позволяющих наблюдать параллельно несколько небесных объектов и изменяющих программу наблюдений по наступлению разного рода событий (изменения погодных условий, наступления небесных событий, например вспышки сверхновой, отказа записывающих устройств). Специфика подобных систем заключается с одной стороны, в сложности динамики каждого элемента системы (процесс слежения за объектом или наблюдательный эксперимент), а с другой — в возможности возникновения разного рода событий, которые по-разному в зависимости от момента их наступления мгновенно изменяют динамику элементов и структуру системы в целом.

Проектирование подобных автоматизированных комплексов, их испытание и выбор оптимальных режимов эксплуатации приводит к специфической постановке ряда инженерных задач и требует новых методов исследования процессов функционирования непрерывно–дискретных систем.

Данная работа посвящена созданию новых методов и алгоритмов моделирования и качественного анализа этого класса динамических систем.

Введем понятие непрерывно-дискретной системы и опишем неформально ее основные свойства.

Непрерывно–дискретной системой называется сложная динамическая система, состоящая из большого числа взаимосвязанных и взаимодействующих элементов различной природы, а именно из элементов, поведение которых описывается непрерывными процессами, имеющими конечную длительность, и элементов, поведение которых описывается дискретными процессами, время выполнения которых несущественно для анализа системы.

Непрерывно–дискретная система содержит в себе в качестве элементов как объекты динамической природы, являющиеся по сути классическими динамическими системами, так и объекты дискретной природы, являющиеся дискретными системами.

Глобальное поведение непрерывно–дискретной системы описывается последовательностью локальных поведений, смена которых происходит под воздействием событий. Наступление того или иного события зависит от значений непрерывных параметров, то есть от функций локальных поведений. С другой стороны, одно событие может порождать другие, и сам дискретный процесс, результатом которого является выбор следующего локального поведения, описывается в общем случае нетривиальным дискретным алгоритмом. Поведение непрерывно–дискретной системы можно, таким образом, представить бесконечной последовательностью сменяющих друг друга сложных длительных непрерывных и сложных мгновенных дискретных поведений (рис.1), а саму систему — графом смены поведений, в котором каждая вершина определяет поведение в текущий момент времени, а каждый переход — условия смены поведений.

Рассмотрим более подробно отдельный элемент системы.

В общем случае поведение элемента может быть представлено тройкой последовательностей:

1. $\{[t_{i-1}, t_i), f_i(t)\}, i = 1, 2, \ldots$ — последовательность локальных поведений, f_i – гладкая вещественная функция;

2. $\{t_i, e_i\}, i = 1, 2, \ldots$ — последовательность событий, приводящих к смене локальных поведений;

3. $\{t_i, a_i\}, i = 1, 2, \ldots$ — последовательность дискретных действий,

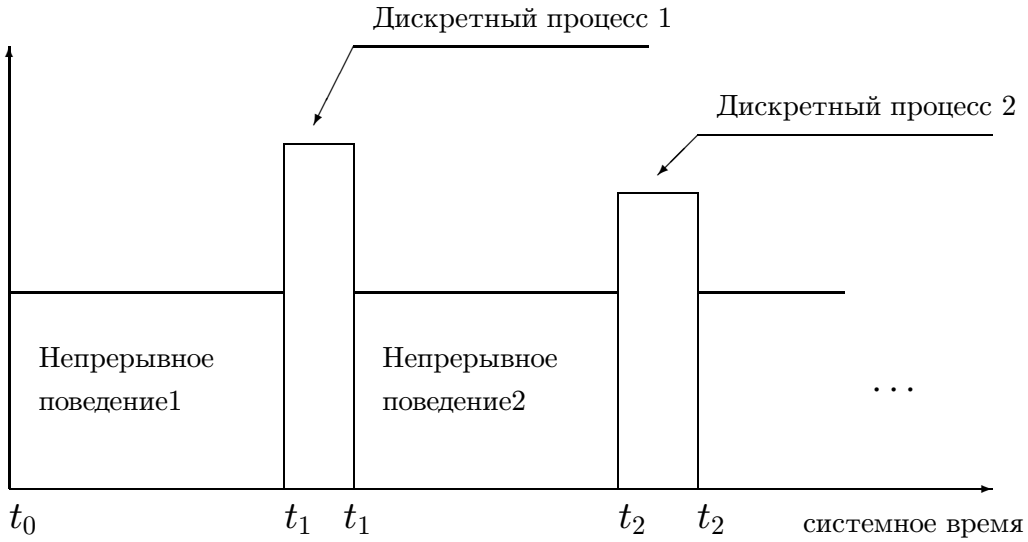


Рис. 1: Схема поведения непрерывно-дискретной системы.

выполняемых при смене локальных поведений.

Локальные поведения $\{f_i\}$ образуют множество F допустимых локальных поведений элемента. Значения $\{e_i\}$ "причин смены поведений" образуют множество E особых событий. Наконец, дискретные действия $\{a_i\}$ образуют множество Act допустимых дискретных действий.

Глобальное поведение элемента — это множество всех возможных поведений при заданных конечных множествах F, E и Act.

Характерными особенностями поведения элемента непрерывно–дискретной системы являются:

• возможность мгновенного изменения значений параметров элемента – разрывы в фазовых траекториях

• возможность мгновенной смены поведения элемента

• выбор нового локального поведения как результат решения (вообще говоря нетривиальной) дискретной задачи с использованием текущей информации от окружающих элементов и в этом смысле недетерминированность поведения каждого отдельно взятого элемента

• возможность периодических движений фазовых траекторий

Отдельные элементы взаимодействуют между собой. Это взаимодействие может носить как непрерывный, так и дискретный характер. Наступление событий может зависеть от значений непрерывных параметров системы, с другой стороны, наступление события приводит к мгновенному изменению значений параметров и к мгновенной

смене поведения одного или нескольких элементов системы. Необходимость учитывать взаимодействие приводит к дальнейшей структуризации описания поведения элемента, а именно в разделении переменных элемента на входные, выходные и переменные состояния. Сами взаимодействия осуществляются по каналам связи, описание которых также должно быть представлено в математической модели системы.

В непрерывно–дискретной системе возможна также иерархическая структура, когда элемент сам является сложным объектом, обладающим свойствами системы.

Наконец, в процессе функционирования непрерывно–дискретной системы возможно изменение ее структуры, поскольку одни элементы могут порождать и уничтожать другие. Например, такая ситуация возникает в системах слежения, когда управляющий элемент следит за поведением элементов различного типа, которые то возникают, то исчезают из его поля зрения.

Таким образом, к основным характеристикам системы в целом следует отнести:

• параллельную композицию элементов как основной принцип конструирования системы

• дискретное и непрерывное взаимодействие между элементами, осуществляемое по каналам связи – необходимость описания модели каналов связи

• возможность иерархической структуры

• возможность мгновенной смены структуры.

Таким образом, можно дать следующее определение модели непрерывно–дискретной системы:

Определение (элемент системы). *Элементом непрерывно–дискретной системы называется совокупность базовых понятий:*

$$E \;=\; \{\; T\;,\;\; Z\;,\;\; X\;,\;\; Y\;,\;\; F\;,\;\; E\;,\;\; Act\;\},$$

где

$T = t_d : N \to R_{\geq 0}$ *— последовательность временных отсчетов*

Z *— множество переменных состояний с заданной областью значений* $D_1 \times \ldots \times D_n$

X *— множество входных переменных (каналов входа)*

Y — *множество выходных переменных (каналов выхода)*

F — *множество допустимых локальных поведений*

E — *множество причин смены поведения (событий)*

Act — *множество допустимых дискретных операций*

Конкретизация основных компонент элемента порождает различные подклассы непрерывно–дискретных систем.

Определение. *Непрерывно–дискретной системой называется пара:*

$$S \quad = \quad \{ \ Time \ , \quad \Sigma \ \},$$

где

$Time = R_{\geq 0}$ — *модель времени*

Σ — *модель объекта,являющегося:*

(i) элементом

(ii) элементом, в котором хотя бы одно локальное поведение заменено на Σ

(iii) параллельной композицией Σ *с определенной на ней моделью каналов связей*

Разработка методов анализа и моделирования непрерывно–дискретных систем имеет уже более чем сорокалетнюю историю [1, 15, 8, 19, 18, 10, 44]. Вслед за созданием метода точечных преобразований для качественного анализа свойств разрывных колебательных систем в теории нелинейных колебаний (А.А.Андронов, Ю.И.Неймарк и др. [1]), которые можно считать первыми изученными динамическими системами с дискретной компонентой, появляются и различные математические модели и формализмы, предназначенные для моделирования поведения непрерывно–дискретных систем. Среди них агрегативные системы Н.П.Бусленко (1963г.) [8], непрерывно–дискретная модель В.М.Глушкова (1973г.) [19] и другие.

Среди современных подходов в области моделирования и анализа непрерывно–дискретных систем следует отметить многочисленные попытки создания средств для компьютерного моделирования и автоматического анализа поведения сложных динамических систем с дискретной компонентой в различных пакетах компьютерного моделирования (Simulink [76], TESS [18], VisSim[1],Statemate MAGNUM[1], i-Logic

[1]информация о системах может быть найдена в международной компьютерной сети InterNet, http://www.vissim.com/vissim.htm

Rhapsody[1]). Наиболее интересным здесь, как нам кажется, является гибридный подход и модель гибридной системы А.Пнуэли (1992г.) [64].

Анализ подходов к моделированию, которые лежат в основе современных вычислительных комплексов, показывает, что сегодня применяются только два подхода к исследованию непрерывно–дискретных систем — представление поведения системы в виде последовательности классических динамических систем и численное "траекторное" моделирование или упрощение непрерывной части и использование технологии дискретного моделирования и анализа. Существующие системы моделирования реализуют, таким образом, лишь ограниченное расширение базовых непрерывных или дискретных моделей и методов.

Между тем, важной отличительной чертой непрерывно–дискретной системы является нетривиальность каждой из ее компонент (как непрерывной, так и дискретной) и упрощение поведения той или иной компоненты приводит к неполноте полученной вычислительной модели и проводимого анализа. При моделировании непрерывно–дискретных систем возникают как проблемы, типичные для классических динамических систем, которые могут быть исследованы средствами непрерывного анализа (к примеру, вопросы устойчивости), так и проблемы, типичные для дискретных параллельных систем, которые можно эффективно решать методами верификации дискретных систем (проблемы тупиков, выявление наличия циклов и пр.).

На сегодняшний день подхода, к котором бы одинаково глубоко исследовались оба аспекта непрерывно–дискретной системы, не существует.

Наблюдая за развитием в области методов исследования непрерывно–дискретных систем, нельзя не заметить, что в последние несколько лет прослеживается тенденция к сближению и попытке совместного использования технологий математического моделирования непрерывных и дискретных процессов.

Становится все более очевидным, что задача автоматизации моделирования и анализа непрерывно–дискретных систем требует со-

здания нового подхода (новой технологии) к моделированию и анализу непрерывно–дискретных систем и разработки методов исследования, интегрированно сочетающих в себе возможности рассмотрения и анализа одной и той же системы, представленной одной моделью, в разных аспектах и объединяющих в себе элементы технологий моделирования как непрерывных, так и дискретных процессов.

В основу нового подхода должен быть положен принцип декомпозиции системы на две равноправные (равнозначные) непрерывную и дискретную компоненты и использование для каждой из них оптимальных инструментов моделирования и анализа.

Высокий уровень современной технологии моделирования динамических систем (работы в области современной теории управления [21, 17], пакеты Simulink [76], Model Vision 3.0 [57]), появление автоматических методов качественного и параметрического анализа дискретных систем, разработанных в теории реактивных систем (работы Ч.Хоара, Л.Лэмпорта, Е.Кларка, А.Пнуэли, Д.Харела, Т.Хензингера, системы VisSim[1], HyTech [51]) предопределяет существование богатой базы для появления такой технологии. Бурное развитие символьного анализа, появление технологии символьных вычислений (пакеты символьных вычислений Mathematica [74], Maple [75]) также оказываются полезными при исследовании сложных и, в частности, непрерывно–дискретных систем.

Однако, следует обратить внимание на то, что области непрерывного и дискретного моделирования на протяжении всей своей истории развития являлись абсолютно независимыми. В каждой из них разработана своя терминология, определены свои цели исследования, при качественном анализе принципиальными и интересными являются разные вопросы.

Создание нового подхода к моделированию и качественному анализу непрерывно–дискретных систем, требует, таким образом:

• введения единой терминологии для дискретной и непрерывной компоненты и разработки новой математической модели непрерывно–дискретной системы;

• обозначения круга вопросов, которые имеют первостепенное значение при моделировании и анализе непрерывно–дискретных систем;

• выбора необходимых оптимальных методов дискретной и непрерывной области в соответствии с поставленными вопросами;

• необходимой модификации, возможного объединения методов для учета влияния компонент друг на друга.

В рамках создания нового подхода к моделированию и качественному анализу непрерывно–дискретных систем автор диссертационного проекта проводит исследования последних достижений в области дискретного моделирования, интересных с точки зрения возможности их включения в такой подход. Основное внимание уделено изучению гибридного направления (современного направления дискретного моделирования непрерывно–дискретных систем) [64, 62, 49, 54, 46], методу символьной верификации [39, 67, 73] и исследованию границ разрешимости гибридных методов [53, 52, 56, 68].

Применение численного моделирования представляется чрезвычайно трудоемкой задачей при анализе поведения и свойств непрерывно–дискретных систем. Кроме того, упускаются из виду некоторые проблемы, которые могут возникать в дискретной компоненте (например, проблемы тупиков, наличие циклов и др.).

Принципиальные моменты гибридного подхода, разработанного как альтернатива численному моделированию, можно сформулировать в следующих пунктах:

— формализация свойств системы, использование специального языка спецификаций свойств поведения (темпоральная логика), что позволяет формулировать сложные качественные и количественные требования к поведению системы;

— использование "поведенческих" моделей как средств (графических и текстуальных) компактного и наглядного описания поведения системы, разбиение пространства состояний на классы эквивалентности относительно исследуемых свойств. Это выводит на новый уровень задачу моделирования: вместо многочисленных экспериментов для построения глобального поведения системы достаточно обойти все пути в графе;

— использование методов автоматической верификации, основанных на достижениях в области символьных вычислений; автоматизация задач качественного параметрического анализа.

Основным отличием и преимуществом гибридного подхода является то, что с его помощью можно с легкостью моделировать и анализировать системы со сложным дискретным управлением вплоть до управления, описываемого программами и конечными автоматами. Теория управления, имея развитый аппарат для исследования сложной динамики окружения в системах управления, не имеет однако средств, позволяющих увеличить сложность дискретных компонент выше определенного предела. Другим преимуществом является возможность моделирования поведения и параметрического анализа систем, в которых характеристики могут быть определены не точно, а заданы, например, в виде интервалов. Таким образом, гибридный подход может рассматриваться в некотором роде и как развитие методов интервального анализа для исследования сложных динамических систем.

Таким образом, привлекательность гибридного подхода по сравнению с численным моделированием может быть кратко сформулирована в следующих пунктах:

а) возможность анализа качественных свойств поведения и оценок областей достижимости системы без траекторного моделирования;

б) возможность исследования вопросов параллельности в непрерывно–дискретных системах;

в) неограниченная сложность дискретной компоненты в исследуемых системах;

г) возможность интервального задания параметров.

С другой стороны, анализ работ по исследованию границ разрешимости гибридных методов позволяет сделать вывод о том, что это направление сегодня не может конкурировать с современными численными компьютерными системами моделирования. Один из возможных путей разрешения этой проблемы по мнению автора работы заключается в значительном расширении набора алгоритмов линейной аппроксимации для нелинейных гибридных автоматов.

Основным результатом работы явилась разработка нового алгоритма линейной аппроксимации нелинейных гибридных систем, основанного на идее совместного использования метода символьной

верификации гибридных систем, численных методов теории ОДУ и методов интервального анализа. Алгоритм обобщенного таймер-преобразования расширяет класс гибридных систем, для которых возможна символьная верификация, до параллельной композиции гибридных автоматов с линейными системами ОДУ и ограниченной интервальной инициализацией. Оценена вычислительная сложность алгоритма, возможные затраты при реализации.

Алгоритм обобщенного таймер-преобразования позволяет расширить возможности различных существующих систем компьютерного моделирования (система автоматической верификации гибридных систем HyTech [51], система дискретного моделирования Covers [7], система моделирования Model Vision [13, 57] и другие). В настоящее время начата реализация алгоритма обобщенного таймер-преобразования в системе компьютерного моделирования Model Vision 3.0.

Метод символьной верификации совместно с алгоритмом обобщенного таймер-преобразования в численных компьютерных системах моделирования может значительно упростить процесс моделирования непрерывно-дискретных систем, так как в этом случае становится возможным проверка качественных свойств поведения системы и параметрический интервальный анализ на предварительном этапе моделирования, а также позволит исследовать вопросы параллелизма и взаимодействия компонент непрерывно-дискретной системы. В компьютерных системах дискретного моделирования, таких как Covers, реализация алгоритма позволит проводить качественный анализ поведения непрерывно–дискретных систем чисто дискретными методами, если выполнены условия приводимости анализируемой системы к системе временных переходов. В компьютерных системах моделирования, основанных на символьной верификации, таких как HyTech, реализация алгоритма значительно расширит класс непрерывно–дискретных систем, для которых возможна верификация.

Диссертационная работа состоит из двух разделов. Первый раздел содержит необходимые сведения о гибридном направлении исследования непрерывно-дискретных систем. Представлены основные математические модели дискретных параллельных систем, базовая

модель гибридного подхода(гибридный автомат) и метод символьной верификации. Во втором разделе представлен алгоритм обобщенного таймер-преобразования. Введен класс гибридных автоматов с линейными системами ОДУ, доказываются теоремы о возможности символьной верификации этого класса гибридных автоматов. Оценивается вычислительная сложность алгоритма обобщенного таймер-преобразования, предложены метод покоординатных оценок и метод оценки временного интервала для решения задачи оценивания фазового состояния линейных систем ОДУ с постоянными коэффициентами. В приложении 1 рассматривается возможность использования символьного подхода для различных математических моделей непрерывно-дискретных систем. В приложении 2 демонстрируется метод покоординатных оценок на математической модели часовых ходов.

Автор работы выражает глубокую благодарность своим научным руководителям профессору Ю.Г.Карпову и Ю.Б.Сениченкову за постоянную поддержку и помощь в работе. Автор работы благодарит профессора С.М.Устинова за ценные замечания по опубликованным работам, а также профессора А.А.Первозванского за ценные замечания по работе и консультации в области теории и систем автоматического управления, которые повлияли на исследования по теме диссертации. Автор глубоко признателен профессору Ф.Л.Черноусько за консультации по своей книге и ценные советы.

1 Гибридное направление исследования непрерывно-дискретных систем

Глава содержит необходимые сведения о гибридном направлении исследования непрерывно–дискретных систем.

Представлена базовая математическая модель дискретной параллельной системы (система переходов), две основные модели дискретных систем реального времени (система временных переходов и таймированный автомат), которые лежат в основе гибридного направления и метода символьной верификации гибридных систем. Представлена классификация требований к поведению дискретных параллельных систем, которые могут быть верифицированы в рамках дискретного темпорального подхода, и базовый язык спецификаций требований к поведению (темпоральная логика).

Далее подробно представлен метод символьной верификации и два алгоритма линейной аппроксимации гибридных автоматов. Рассмотрены примеры верификации и линейной аппроксимации гибридных автоматов, описаны недостатки существующих алгоритмов линейной аппроксимации.

1.1 Дискретный темпоральный подход

Рассматриваемый в настоящей работе гибридный подход к исследованию непрерывно-дискретных систем является развитием дискретного темпорального подхода, развитого в работах А.Пнуэли, Д.Харела, Е.Кларка, Е.Эмерсона, З.Манны и других современных авторов [29, 37, 41, 44, 43, 63] для решения задач моделирования и анализа дискретных параллельных и распределенных систем.

Элементами дискретной параллельной системы являются дискретные системы (называемые процессами), параллельно функционирующие в дискретном времени. Параллелизм является отражением либо *физического* (*аппаратного*) параллелизма — одновременной работы нескольких физических устройств, либо *логического* параллелизма — концептуального параллелизма, введенного человеком, независимо от того, происходит ли работа процессов последовательно или параллельно. Процессы-участники параллельной системы могут быть независимыми друг от друга и конкурировать за общие ресурсы (асинхронные конкурентные системы), обмениваться информацией, выполняя общую задачу (кооперативные распределенные системы), предоставлять распределенный доступ к общим данным (распределенные базы данных), иметь сложную динамическую и иерархическую структуру взаимодействий (вычислительные сети), иметь жесткие ограничения на время выполнения отдельных операций (параллельные си-

стемы реального времени).

В дискретных параллельных системах возникает необходимость обеспечивать взаимодействие параллельных процессов: возникают проблемы критических секций, тупиков и зацикливаний, проблемы разработки синхронизирующих примитивов, организации очередей, защиты ресурсов и многие другие.

Дискретный темпоральный подход включает в себя четыре основные компоненты:

- Базовую математическую модель параллельной системы и ее эффективное представление в ЭВМ (вычислительную модель). Математическая модель (или семантика) описывает взаимодействие и совместное поведение параллельных компонент системы в дискретном времени. Модели параллельных систем разделяются на три основных типа, которые отличаются друг от друга различными способами представления параллелизма (и порождаемых им недетерминизма и конкурентности) в параллельных системах. Здесь мы ограничимся их перечислением, подробно см. [63]:

  – Представление поведения параллельной системы множеством бесконечных цепочек состояний, последовательно принимаемых системой в дискретном времени, в которых переход к следующему состоянию соответствует одному последовательному действию некоторой компоненты параллельной системы (линейная или "интерливинговая" семантика).

  – Представление поведения параллельной системы бесконечным деревом состояний, в котором узлы соответствуют состояниям, принимаемым системой в дискретном времени, а точки ветвления соответствуют недетерминированному выбору нового состояния в одной компоненте или между параллельными компонентами (ветвящаяся семантика).

  – Представление поведения параллельной системы сетью узлов состояний, принимаемых системой в дискретном времени, на которой определены отношения "частичного порядка" и "конкурентности" между последовательными состояниями ее параллельных компонент (семантика частичного порядка).

В настоящей работе мы будем использовать линейную и ветвящуюся семантические модели, семантика частичного порядка используется в Сетях Петри [36].

- Язык спецификаций, с помощью которого могут быть формализованы свойства, которым должна удовлетворять параллельная система. Каждое свойство, описанное на языке спецификаций (в виде формулы), должно выполняться на множестве всех поведений или, как принято говорить в этой области, на семантике ее модели. Таким образом, семантика рассматривается как models (в логическом смысле) для формул, то есть определены правила для проверки истинности формулы на заданной семантике. Множество всех формально описанных свойств, которым должна удовлетворять параллельная система, называется ее спецификацией.

- Классификацию свойств. Все свойства поведения, которые могут быть формально описаны на языке спецификаций, могут быть разбиты на классы. Для каждого класса можно применять свои правила доказательства соответствия семантики заданным свойствам.

- Теорию верификации, включающую в себя развитие различных подходов к доказательству соответствия вычислительной модели и ее спецификации. Среди подходов к верификации выделяются аксиоматическая система доказательств, базирующаяся на темпоральной логике, [63, 64], алгоритмы автоматической верификации [29, 41, 42, 43], алгоритмы параметрического анализа [7, 70].

При переходе от дискретных параллельных систем и систем реального времени к более сложному классу гибридных систем, в которых участвуют непрерывные процессы, каждая из компонент дискретного темпорального подхода должна быть определенным образом модифицирована.

В настоящей главе представлены модели и методы верификации дискретного темпорального и гибридного подходов, которые используются в работе.

1.2 Математические модели дискретного темпорального подхода

Базовой математической моделью в дискретном темпоральном подходе является модель Систем Переходов А.Пнуэли [63]. Опишем семантику и синтаксис этой модели.

Линейная модель. Пусть дан ориентированный мультиграф $G = \{ S , E \}$, в котором $S = \{s_1, \ldots, s_p\}$ – конечное множество вершин (состояний), $E = \{e_{ij}^k = (s_i, s_j), k = 1 \ldots, q\}$ – конечное множество дуг (переходов).

Введем множество переменных состояния X с заданной областью значений $X \in D$ (например, $D \subseteq R^n$). Каждой дуге $e \in E$ поставим в соответствие пара $(P_e(X), f_e(X))$, где $P_e(X) : D \to \{true, false\}$ – условие срабатывания перехода e (логическая формула над переменными состояния), $f_e(X) : D \to D$ – действие перехода e (формула преобразования переменных состояния на дуге).

Одна (или более) вершина помечается как начальная; приписываемое ей значение вектора переменных состояния $X_0 \in D$ (или область $D_0 \subset D$) называется начальным значением (или множеством начальных значений) системы переходов. Множество начальных состояний обозначается Θ.

Последовательность $T_d = \{i\}_0^\infty$ называется дискретным временем.

В каждый момент дискретного времени i система переходов находится в одном из состояний $s^{(i)} \in S$, которое называется текущим. Соответствующее этому состоянию значение вектора переменных X обозначается X_i. В каждый следующий момент времени текущим состоянием может стать только состояние $s' \in S$, связанное переходом с текущим состоянием: $\exists e = (s^{(i)}, s') \in E$. Из множества претендентов на новое текущее состояние $\{s', \exists e = (s^{(i)}, s') \in E\}$ выбирается то, для которого выполнено условие срабатывания соответствующего перехода: $s^{(i+1)} = s',\ P_{(s^{(i)},s')}(X_i) = true$. Переходы из текущего состояния, для которых условия срабатывания истинны, называются возможными (или активизированными) переходами, их множество обозначается $Enabled(s^{(i)})$, соответствующее этим переходам множество значений переменных состояний $X = f_{e \in Enabled}(X_i)$ называется множеством до-

стижимости текущего состояния и обозначается $Reach(s^{(i)})$. Выбор нового текущего состояния осуществляется выбором одного из возможных переходов, который называется "сработавшим" переходом.

Система переходов называется детерминированной, если для любого i множество $Enabled(s^{(i)})$ содержит ровно один переход.

Введем конечное множество меток L и определим отображение $syn : E \to L$, которое каждой дуге $e \in E$ мультиграфа G ставит в соответствие метку $a \in L$. Множество L называется алфавитом системы переходов.

Вычислением (линейным вычислением) системы переходов назовем последовательность $\sigma = \{(s^{(i)}, X_i)\}_{i=0}^{\infty}$, где

a) $s_0 \in \Theta, \quad X_0 \in D_0$, и

b) $(\forall\ i)(\exists\ e \in E) : e = (s^{(i)}, s^{(i+1)}), P_e(X_i) = true, X_{i+1} = f_e(X_i)$.

Множество всевозможных вычислений, отвечающих множеству начальных состояний, назовем поведением системы переходов. Таким образом определяется линейная семантика (линейная модель поведения) дискретной параллельной системы.

Древовидная модель. Рассмотрим теперь другую модель представления поведения дискретной параллельной системы.

Вернемся к рассмотрению мультиграфа $G = \{S, E\}$. Определим множество переменных состояния X, конечный алфавит меток переходов L, пару $(P(X), f(X))$ для каждой дуги мультиграфа и выделим начальные состояния Θ и множество начальных значений D_0.

В каждый дискретный момент времени i будем рассматривать множество состояний $\{s^{(i)} \in S\}$, включающее в себя состояния системы переходов, которые могут стать текущими в момент времени i.

Будем строить дерево, в котором корню сопоставляется начальная вершина графа G, каждому узлу i-того уровня сопоставляется состояние, которое может стать текущим в дискретный момент времени i, а множество потомков каждого узла включает в себя всех претендентов на новые текущие состояния (связанных переходами с этим текущим состоянием $\exists e = (s^{(i)}, s') \in E$), для которых выполнены условия срабатывания соответствующих переходов: $s^{(i+1)} = s',\ P_{(s^{(i)}, s')}(X_i) = true$.

Каждому узлу поставим в соответствие значение вектора переменных X_i, отвечающее состоянию $s^{(i)}$.

Каждому пути в таком дереве будет соответствовать возможная цепочка состояний, принимаемых последовательно системой переходов в дискретные моменты времени. Точки ветвления будут соответствовать состояниям, в которых реализуется недетерминированный выбор нового состояния из множества возможных (таким образом, могут быть идентифицированы состояния дискретной параллельной системы, в которых осуществляется недетерминированный выбор следующего состояния).

Назовем это (возможно бесконечное) ветвящееся дерево деревом вычислений системы переходов G.

Система переходов называется детерминированной, если каждое его дерево вычислений вырождается в последовательность.

Таким образом определяется древовидная семантика (древовидная модель поведения) дискретной параллельной системы с выделением состояний, в которых осуществляется недетерминированный выбор между несколькими альтернативными поведениями.

Линейную и древовидную модели поведения дискретной параллельной системы можно описать следующим образом:

Определение 1.2.1 *Системой переходов называется кортеж:*

$$\mathcal{S} = \{ X,\ S,\ E,\ L,\ \Theta,\ P,\ F,\ syn \},\ где \tag{1}$$

X – конечное множество переменных состояния с заданной областью определения D;

S – конечное множество состояний;

$\Theta \subset S$ – множество начальных состояний;

L – конечный алфавит системы переходов;

P – конечный набор логических функций над переменными состояния;

F – конечный набор формул преобразования переменных состояния;

$syn : E \to L$ – заданное конечное отображение;

$E \subseteq \{S \to S\}$ – конечное множество переходов, где каждому переходу e поставлены в соответствие условие перехода P_e, действие перехода f_e и метка $a = syn(e)$;

Конкретизация компонент описания системы переходов порождает различные модели поведения дискретной параллельной системы.

Для исследования взаимодействия компонент в дискретных параллельных системах в дискретном темпоральном подходе используется композиционный принцип. Его суть заключается в построении одной модели, которая описывает совместное поведение параллельно функционирующих и взаимодействующих друг с другом дискретных компонент. Для этих целей вводится понятие параллельной композиции систем переходов.

Пусть $A_1 = \{X_1, S_1, E_1, L_1, \Theta_1, P_1, F_1, syn_1\}$ и $A_2 = \{X_2, S_2, E_2, L_2, \Theta_2, P_2, F_2, syn_2\}$ – две системы переходов.

Определение 1.2.2 *Параллельной композицией двух систем переходов A_1 и A_2 называется система переходов*

$$A = A_1 \times A_2 = \{\ X,\ S_1 \times S_2,\ L,\ \Theta_1 \times \Theta_2,\ E,\ P,\ F,\ syn\ \},$$

в которой

$X = X_1 \cup X_2$;

$L = L_1 \cup L_2$;

E содержит переход $e = ((s_1, s_2), (s_1', s_2'))$ тогда и только тогда, когда

(1) $e_1 = (s_1, s_1') \in E_1 \ \wedge\ s_2 = s_2' \ \wedge\ L_2 \cap syn(e_1) = \emptyset$, *или*

(2) $e_2 = (s_2, s_2') \in E_2 \ \wedge\ s_1 = s_1' \ \wedge\ L_1 \cap syn(e_2) = \emptyset$, *или*

(3) $e_1 = (s_1, s_1') \in E_1 \wedge e_2 = (s_2, s_2') \in E_2 \wedge L_1 \cap syn(e_2) = L_2 \cap syn(e_1)$,

причем,

для случая (1) $P_e = P_{e_1}, f_e = f_{e_1}, syn(e) = syn(e_1)$;

для случая (2) $P_e = P_{e_2}, f_e = fe_2, syn(e) = syn(e_2)$;

для случая (3) $P_e = P_{e_1} \wedge P_{e_2}, f_e = f_{e_1} \wedge f_{e_2}, syn(e) = syn(e_1) \cup syn(e_2)$.

Состоянием параллельной композиции систем переходов, таким образом, является кортеж, содержащий по одному состоянию каждой ее компоненты. Переходы в параллельной композиции определяются в соответствии со следующим принципом: если обе системы имеют общую метку $a \in L_1 \cap L_2$, то каждый a-переход e_1 в системе переходов A_1 должен произойти одновременно с a-переходом e_2 в системе переходов

A_2, то есть в параллельной композиции A эти переходы объединяются в единственный переход с меткой a с конъюнкцией условий переходов e_1 и e_2 и совместным выполнением действий над переменными состояний e_1 и e_2 систем переходов A_1 и A_2. Остальные переходы в параллельной композиции, не принадлежащие $L_1 \cap L_2$, наследуются из соответствующих компонент. Этот принцип называется механизмом синхронизирующих меток.

Система переходов не несет информации о длительности последовательных действий отдельных процессов в параллельной системе. В дискретном темпоральном подходе разработаны два основных способа введения непрерывного времени в модель параллельной системы. Они порождают две основные модели для спецификации дискретных систем реального времени – систему временных переходов и таймированный автомат.

Система временных переходов является расширением системы переходов, в которую дополнительно введены задержки срабатывания переходов [7].

Здесь нам понадобится понятие интервала, под которым понимается отрезок вещественной оси $I = [a, b], 0 \leq a \leq b, a, b \in R$. Множество всевозможных интервалов обозначим через $\mathcal{I}(R)$.

Определим систему временных переходов следующим образом:

Определение 1.2.3 *Системой временных переходов называется тройка:*

$$\mathcal{S}^T = < \mathcal{S}, L, U >, \text{ где} \tag{2}$$

$\mathcal{S}$ — *система переходов (1)*
$L = \{l_\tau, \tau \in \mathcal{T}\}$
$U = \{u_\tau, \tau \in \mathcal{T}\}$
$0 \leq l_\tau \leq u_\tau, \quad [l_\tau, u_\tau] \in \mathcal{I}(R)$
Для каждого $\tau \in \mathcal{T}$ *величины* l_τ, u_τ *представляют собой минимальную и максимальную задержку срабатывания перехода* τ*, если он активизирован.*

Поведение системы временных переходов должно удовлетворять следующим свойствам.

Каждой дуге системы переходов поставим в соответствие интервал $I_k = [l_k, u_k]$, который называется временным интервалом задержки срабатывания перехода. Значения l_k, u_k называются минимальной и максимальной задержкой срабатывания перехода соответственно. Семантику системы временных переходов определим как множество ее вычислений, каждое из которых является последовательностью $\rho^T = \{(s_i, X_i, t_i)\}_{i=0}^{\infty}$, удовлетворяющей следующим свойствам:

a) последовательность $\{(s_i, X_i)\}_{i=0}^{\infty}$ является вычислением в соответствующей системе переходов S;

b) последовательность дискретных моментов времени $t_i \in T$ упорядочена;

c) $\forall e \in E$ и $j \geq 0, \exists i \leq j$, т.ч. $t_i + l_e \leq t_j$ и переход e непрерывно активен в состояниях $s_i \ldots s_j$ но еще не сработал;

d) $\forall e \in E$ и $i \geq 0, \exists j, i \leq j$, т.ч. $t_i + u_e \geq t_j$ и переход e сработал в j или стал неактивен.

При значениях $l_k = u_k = 0$ на всех переходах система временных переходов вырождается в систему переходов.

Альтернативный способ введения непрерывного времени в модель параллельной системы реализован в модели таймированного автомата.

Рассмотрим конечный автомат $A = \{\Sigma, V, V_0, E\}$, где

V – конечное множество вершин (или локаций);

E – конечное множество дуг;

$V_0 \subset V$ – множество начальных вершин;

Σ – конечный входной алфавит, и определено отображение $syn : E \to \Sigma$. Каждой дуге автомата поставлен в соответствие символ входного алфавита.

Введем модель непрерывного времени $T = R_{\geq 0}$.

Назовем "локальными часами" (или локальным таймером) вещественную кусочно-линейную вещественную функцию $x(t)$ с областью значений $D(x) = R_{\geq 0}$. Разобьем T последовательностью точек $\theta : t_0, t_1, \ldots, t_i, \ldots$ на отрезки $[t_0, t_1], [t_1, t_2], \ldots$. На каждом из отрезков $t \in [t_i, t_{i+1}]$ $x(t) = t - t_0$, точка t_i называется моментом "перезапуска таймера".

Введем в автомат A конечное множество локальных часов X.

Каждой дуге автомата $e \in E$ поставим в соответствие пару $(pre_e(X), post_e(X))$, описывающих продвижение времени и перезапуск таймеров на дуге. $pre_e : D(X) \to \{true, false\}$ – логическая формула, описывающая условие перехода по дуге автомата, $post_e : D(X) \to D(X)$ – набор перезапусков некоторых часов из множества X на дуге.

В каждый момент времени $t \in T$ таймированный автомат находится в одной из локаций $v \in V$, которая называется текущей. Поведение таймированного автомата во времени представляется последовательностью фаз двух типов:

- "временная" фаза – увеличение системного времени на Δt внутри текущей локации с параллельным увеличением локального времени всех таймеров:

$$\frac{\mathrm{X_0 = X(t_i),\ \ \exists \Delta t \exists t_{i+1} : t_{i+1} = t_i + \Delta t,\ \ \forall t' \in [t_i, t_{i+1}]\ \ X = X_0 + t'}}{\mathrm{(v, X_0) \to^{\Delta t} (v, X)}}$$

- "дискретная" фаза – один (или более) переход по дуге автомата при фиксированном времени с запуском некоторых таймеров:

$$\frac{\mathrm{e = \{v, a, pre, post, v'\} \in E,\ \ pre_e(X_0) = True,\ \ X_1 = post_e(X_0)}}{\mathrm{(v, X_0) \to^{e} (v', X_1)}}$$

Линейную семантику таймированного автомата определим как множество всевозможных пробегов, каждый из которых является последовательностью $\{(v_i, \Delta_i t)\}_{i=0}^{\infty}$, удовлетворяющей условиям:

а) $v_0 \in V_0$;

б) $\forall i \geq 0,\ \ \exists e = (v_i, v_{i+1}) \in E$ и $\Delta_i t$ – длительность временной фазы в локации v_i.

Каждой дуге автомата можно теперь поставить в соответствие не только символ входного алфавита, но и момент времени $t_i \in T$, который соответствует переходу по этой дуге (абсолютному времени дискретной фазы) и вычисляется по формуле $t_{i+1} = t_1 + \Delta t$. Пару (a, t_i) называют событием[2] автомата ($a_i \in \Sigma$ – "суть события" или "класс событий", $t_i \in \theta$ – "момент наступления события").

[2]Входной алфавит Σ называется также алфавитом событий таймированного автомата.

Можно рассматривать и другую, древовидную, модель поведения параллельной системы реального времени и ввести понятие дерева пробегов. Таймированный автомат будем описывать следующим образом [28]:

Определение 1.2.4 *Таймированным автоматом называется кортеж:*

$$A = \{ \Sigma,\ X,\ V,\ V_0,\ V_F,\ E,\ pre,\ post,\ syn \},\ где \tag{3}$$

Σ — *конечный входной алфавит;*

X — *конечное множество локальных таймеров;*

V — *множество вершин автомата (локаций);*

E — *конечное множество дуг;*

pre – *конечный набор логических формул над множеством таймеров;*

$post$ – *набор перезапусков таймеров;*

$syn : E \to \Sigma$ – *заданное конечное отображение;*

$V_0, V_F \subseteq V$ — *начальные и конечные локации, начальным вершинам автомата соответствует* $t = 0$ *и множество начальных значений таймеров* $X_0 = 0$*;*

Каждая дуга $e = (v, a, pre, post, v') \in E$, *где* $v, v' \in V$ – *исходная и целевая локации для дуги* e, *помечена буквой алфавита* $a = syn(e)$, *условием перехода по дуге вида* $pre_e(X) : \wedge_i \sum_{k=1}^{n} h_i x_i \sim c,\ h_i, c \in R, x_i \in X, \sim \in \{<, \leq, =, \geq, >\}$ *и действием на дуге вида* $post_e(X) : \wedge_i x_i = 0, x_i \in X$.

Конкретизация компонент описания таймированного автомата порождает различные модели поведения параллельной системы реального времени.

Если определить соответствие между символами входного алфавита и операциями запуска таймеров так, чтобы каждый введенный таймер отслеживал некоторое временное требование к абсолютному времени наступления отдельных событий или к относительному времени между событиями одного или различных классов, то тогда модель таймированного автомата будет позволять исследовать возможные временные задержки между событиями в описанной модели.

Размерностью таймированного автомата называется число введенных переменных-таймеров.

Параллельная композиция таймированных автоматов может быть определена по аналогии с определением 1.2.2. Параллельная композиция двух таймированных автоматов является таймированным автоматом.

Конечная цепочка пар $\{(a_i, t_i)\}_{i=0}^{n}$, где $a_i \in \Sigma, t_i \in \theta$, а $\theta : t_0, t_1, \ldots, t_i, \ldots$ – определенная нами временная последовательность $(t_0 = 0, \forall i \geq 0, t_i \leq t_{i+1})$, допущенная автоматом, называется таймированным словом автомата. Все слова, которым соответствуют допущенные таймированные слова, образуют язык таймированного автомата. Таймированным ω-словом называется бесконечная цепочка пар $\{(a_i, t_i)\}_{i=0}^{\infty}$, где $a_i \in \Sigma, t_i \in \theta$, допущенная автоматом. Если при этом соответствующая временная последовательность θ неограничена, то такое слово называется расходящимся ω-словом. Все ω-слова, которым соответствуют расходящиеся таймированные ω-слова, образуют ω-язык таймированного автомата, который будем обозначать $Lang(A)$. Язык таймированного автомата называется регулярным, если все его слова являются регулярными выражениями.

Введем теперь понятие гибридного автомата [33].

Назовем "фазовыми переменными" множество вещественных переменных X с областями значений $D(x_k) = R$, поведение которых в непрерывном времени $T = R_{\geq 0}$ описывается кусочно-непрерывными функциями. Разобьем T последовательностью точек $\theta : t_0, t_1, \ldots, t_i, \ldots$ на отрезки $[t_0, t_1], [t_1, t_2], \ldots$. В начальный момент времени $x_k(0) \in D(x_k)$, на каждом отрезке непрерывного времени $[t_i, t_{i+1}]$ задается начальное значение переменной $x_{0k} \in D(x_k)$ и ее непрерывное локальное поведение. Мгновенное изменение значения фазовой переменной при смене локального поведения назовем "мгновенным действием" (или локальной инициализацией переменной).

Гибридный автомат является расширением таймированного автомата.

Введем в таймированный автомат n фазовых переменных.

В каждую локацию автомата поместим n функций локальных поведений, по одной для каждой фазовой переменной Будем называть

эти множества функциями локального поведения вектора фазовых переменных и обозначать $flow(v, X)$. Определим также для каждой локации логическую функцию $inv(v)$, описывающую дополнительные ограничения на значения фазовых переменных в v. Каждой дуге автомата $e \in E$ в свою очередь поставим в соответствие пару $(pre_e(X), post_e(X))$, таких что $pre_e : D(X) \to \{true, false\}$ – набор ограничений на значения переменных X (логическая формула над X), описывающих условие смены локального поведения и перехода по соответствующей дуге автомата, $post_e : D(X) \to D(X)$ – набор мгновенных действий (инициализаций) при смене локального поведения и переходе по дуге.

В каждый момент времени $t \in T$ гибридный автомат находится в одной из локаций $v \in V$, которая называется текущей. Для текущей локации определяется множество возможных начальных значений фазовых переменных $\{X_0(v)\}$. Текущим состоянием гибридного автомата называется пара (v, X), где $v \in V$ – текущая локация, $X \in D(X)$ – некоторая точка локального поведения вектора фазовых переменных, отвечающего локации v.

Поведение гибридного автомата во времени представляется последовательностью фаз двух типов:

- "непрерывная" фаза – увеличение системного времени на Δt внутри текущей локации с преобразованием значений фазовых переменных:

$$\frac{\mathrm{X_0(v) \in inv(v), \exists \Delta t \exists t_{i+1} : t_{i+1} = t_i + \Delta t, \forall t' \in [t_i, t_{i+1}]\ X \in flow(v, X)[X_0][t']}}{\mathrm{(v, X_0) \to^{\Delta t} (v, X)}}$$

- "дискретная" фаза – один (или более) переход по дуге автомата при фиксированном времени с инициализацией некоторых фазовых переменных:

$$\frac{\mathrm{e = \{v, a, pre, post, v'\} \in E,\ \ pre_e(X_0) = True,\ \ X_1 = post_e(X_0)}}{\mathrm{(v, X_0) \to^{e} (v', X_1)}}$$

Совокупность всех дуг, выходящих из текущей локации v, для которых условия $pre(X)$ истины, называются множеством возможных дискретных фаз локации v, а объединение множеств значений фазовых переменных $X = \{post(X')\}$, где $X' \in flow(v)[X_0(v)]$, на всех возможных дугах называется множеством достижимости (или множеством достижимых состояний) этой локации.

Пробегом гибридного автомата назовем последовательность
$\{(v_i, \Delta_i t, X_i)\}_{i=0}^{\infty}$, т.ч.:

а) $(v_0, X_0) \in init$;

б) $\forall i \geq 0, \exists e = (v_i, v_{i+1}) \in E$ и $\Delta_i t$ – длительность непрерывной фазы в локации v_i;

в) $\forall i \geq 0, \quad X_{i+1}(0) \in post_e(X_i(\Delta_i))$.

Линейную семантику гибридного автомата определим как множество его всевозможных пробегов. Как и с предыдущем случае может быть определена и древовидная модель поведения.

Объединение всех траекторий фазовых переменных, соответствующих поведению гибридного автомата, будем называть характеристическим множеством или глобальным множеством достижимости гибридного автомата и обозначать $Reach(H)$.

В определении гибридного автомата нам понадобится понятие предиката, под которым подразумевается логическая формула исчисления предикатов первого порядка с использованием имен переменных и их первых производных и стандартных операций и отношений над заданным числовым полем. В общем случае все предикаты гибридного автомата есть логические формулы над $X \cup \dot{X}$, в качестве числового поля задается множество вещественных чисел R.

Определение 1.2.5 *Гибридным автоматом называется кортеж:*

$$H = \{ \Sigma, V, X, E, flow, pre, post, inv, init, syn \}, \text{ где} \qquad (4)$$

Σ — *конечный входной алфавит (алфавит событий);*

V — *конечное множество вершин автомата (локаций);*

E — *конечное множество дуг;*

X — *конечное множество фазовых переменных;*

pre – конечный набор логических формул над множеством фазовых переменных;

post – конечный набор мгновенных действий над фазовыми переменными;

$syn : E \to \Sigma$ *– заданное конечное отображение;*

$flow$ *— конечное множество локальных поведений фазовых переменных;*

init – предикат, описывающий множество начальных локаций и области начальных значений фазовых переменных;

inv — конечное множество логических функций (предикатов-инвариантов), описывающих дополнительные ограничения на значения фазовых переменных, каждая локация автомата помечена предикатом-инвариантом $inv(v)$*;*

Каждая дуга $e \in E$ *помечена буквой алфавита событий* $a = syn(e)$*, условием перехода по дуге* $pre(e)$ *и множеством мгновенных действий на дуге (инициализацией на дуге)* $post(e)$ *.*

Конкретизация компонент описания гибридного автомата порождает различные модели поведения непрерывно-дискретной системы.

Размерностью гибридного автомата называется число введенных фазовых переменных.

Аналогично с таймированным автоматом, для гибридного автомата определяются понятия языка и ω-языка.

Пусть теперь $H_1 = \{\Sigma_1, V_1, X_1, E_1, flow_1, pre_1, post_1, inv_1, init_1, syn_1\}$ и $H_2 = \{\Sigma_2, V_2, X_2, E_2, flow_2, pre_2, post_2, inv_2, init_2, syn_2\}$ – два гибридных автомата. Определим их параллельную композицию [33].

Определение 1.2.6 *Параллельной композицией двух гибридных автоматов* H_1 *и* H_2 *называется гибридный автомат:*

$$H = H_1 \times H_2 = \{\ X,\ V_1 \times V_2,\ \Sigma,\ E,\ flow,\ pre,\ post,\ inv,\ init,\ syn\ \},$$

в котором

$X = X_1 \cup X_2$*;*

$\Sigma = \Sigma_1 \cup \Sigma_2$*;*

Каждая локация $(v_1, v_2) \in V_1 \times V_2$ *имеет инвариант* $inv(v, v') = inv_1(v) \wedge inv_2(v')$ *и функцию локального поведения* $flow(v, v', X) = flow_1(v, X_1) \wedge flow_2(v', X_2)$*;*

E содержит переход $e = ((s_1, s_2), (s_1', s_2'))$ *тогда и только тогда, когда*

(1) $e_1 = (s_1, s_1') \in E_1 \ \wedge \ s_2 = s_2' \ \wedge \ \Sigma_2 \cap syn(e_1) = \emptyset$, *или*

(2) $e_2 = (s_2, s_2') \in E_2 \ \wedge \ s_1 = s_1' \ \wedge \ \Sigma_1 \cap syn(e_2) = \emptyset$, *или*

(3) $e_1 = (s_1, s_1') \in E_1 \ \wedge \ e_2 = (s_2, s_2') \in E_2 \ \wedge \ \Sigma_1 \cap syn(e_2) = \Sigma_2 \cap syn(e_1)$,

причем, для случая (1) $pre_e = pre_{e_1}, post_e = post_{e_1}, syn(e) = syn(e_1)$;

для случая (2) $pre_e = pre_{e_2}, post_e = post_{e_2}, syn(e) = syn(e_2)$;

для случая (3) $pre_e = pre_{e_1} \wedge pre_{e_2}, post_e = post_{e_1} \wedge post_{e_2}, syn(e) = syn(e_1) \cup syn(e_2)$.

Параллельная композиция гибридных автоматов называется *гибридной системой*. Поскольку автоматы могут иметь общие фазовые переменные, то размерность параллельной композиции лежит в пределах $max(n_1, n_2)$ и $n_1 + n_2$.

Гибридные автоматы принято классифицировать по виду предикатов.

Линейным гибридным автоматом называется такой гибридный автомат, в котором все компоненты представляются в виде линейных предикатов (включая функции локальных поведений $flow$ и мгновенные действия $post$), то есть являются логической комбинацией линейных термов, истинность которых соответствует выполнению линейных неравенств вида $\sum_{i=1}^{k} a_i y_i \sim c$, где $a_i, c \in R, y \in X \cup \dot{X}, \sim \in \{<, \leq, =, \geq, >\}$ [48].

Линейным гибридным автоматом с постоянными целочисленными коэффициентами называется линейный гибридный автомат, в котором функции локальных поведений $flow$ имеют вид $\dot{x} = K$, где $K \in Z$, мгновенные действия $post$ являются логической комбинацией термов вида $x \in [a, b], a, b \in Z, x \in X$, а все другие компоненты описания имеют вид линейных предикатов с целочисленными константами [69].

Введем понятие прямоугольного дифференциального включения. Говорят, что поведение вещественной переменной x удовлетворяет прямоугольному дифференциальному включению, если ее значение меняется непрерывно и существуют такие $L, U \in R$, что $L \leq U$ и

$\dot{x} \in [L, U]$.

Рациональным интервалом назовем интервал $I = [a, b]$, где $a, b \in Q$. Интервал называется ограниченным, если a, b – конечные числа.

Прямоугольным автоматом называется гибридный автомат, в котором локальные поведения описаны в виде прямоугольных дифференциальных включений для каждой переменной $x \in X$, то есть имеют вид $flow(v, x) : \dot{x} \in I$, где I – рациональный (возможно неограниченный) интервал, функции преобразования переменных на дугах *post* имеют вид рациональных (возможно неограниченных) интервалов, а другие предикаты автомата $(pre, inv, init)$ описывают в пространстве Q^n прямоугольные области, то есть являются декартовым произведением рациональных интервалов (возможно неограниченных) [53]. Внутри каждой локации v значение каждой переменной $x \in X$ прямоугольного автомата меняется из некоторой точки x_0 до некоторой другой точки x_f, где $x_0 \in post(e(.., v))[x], pre(e(v, ..))[x_f] = true$, по любой непрерывной траектории, удовлетворяющей предикату-инварианту локации $inv(v)$ и дифференциальному включению $flow(v, x)$.

Множество переменных автомата, которые участвуют в функции $post_e$ на дуге e, называется множеством инициализации дуги и обозначается $jump(e)$.

Прямоугольный автомат называется *инициализированным*, если для каждой переменной x_i и для каждой дуги автомата $e(v_1, v_2)$ выполняется условие: если $flow(v_1)_i \neq flow(v_2)_i$, тогда $x_i \in jump(e)$, то есть если в локациях v_1 и v_2 функции локального поведения переменной x_i на совпадают и существует дуга $e = (v_1, v_2)$, то на ней переменная должна быть обязательно инициализирована.

Прямоугольный автомат назовем автоматом *с точечной инициализацией*[3], если: 1) множество $init(v)$ для каждой локации v либо пусто, либо определяет точку $X \in D(X)$ в фазовом пространстве автомата; 2) на любой дуге e инициализация переменных из множества $jump(e)$ задается точкой в фазовом пространстве автомата.

Прямоугольный автомат назовем автоматом *с ограниченной ин-*

[3]В дискретном темпоральном подходе такой автомат называется автоматом с детерминированной инициализацией.

тервальной инициализацией[4], если: 1) область фазового пространства, соответствующая истинности начального предиката $init$, ограничена и все интервалы в описании локальных поведений ограничены; 2) инициализация для множества $jump(e)$ на каждой дуге e автомата описывается ограниченным интервалом.

Наконец, если все фазовые переменные гибридного автомата являются таймерами, то гибридный автомат вырождается в таймированный автомат.

1.3 Классификация и спецификация свойств поведения в дискретном темпоральном подходе, темпоральная логика

В разделе 1.2 были рассмотрены модели параллельных систем. В данном разделе представлены две другие составляющие компоненты дискретного темпорального подхода — классификация свойств поведения модели и язык спецификаций этих свойств (для линейных и древовидных моделей).

Пусть Σ – множество состояний вычислительной модели. Обозначим через Σ^+ – множество всех конечных непустых последовательностей состояний, через Σ^ω – множество всех бесконечных последовательностей состояний, а через $\Sigma^\infty = \Sigma^+ \cup \Sigma^\omega$ – множество всех последовательностей состояний.

Очевидно, что поведение заданной модели всегда является подмножеством Σ^∞. Набор ограничений, выделяющий это подмножество из всего множества последовательностей Σ^∞, будем называть свойствами поведения или требованиями к поведению.

Требования к поведению могут носить как качественный характер – к таким требованиям относятся принципиальная возможность или невозможность присутствия конкретных состояний из множества состояний Σ в вычислениях, порядок следования и причинно-следственные связи между отдельными состояниями (будем называть их качественными требованиями к поведению), — так и представлять

[4]В дискретном темпоральном подходе такой автомат называется автоматом с ограниченным недетерминизмом.

собой временные требования к моментам появления конкретных состояний в последовательностях состояний.

Требования к поведению отражают реальные требования к параллельным системам: например, гарантию недопустимости определенных (аварийных) ситуаций в процессе функционирования системы, требование отсутствия тупиковых ситуаций при работе с общими ресурсами, требование взаимного исключения при вхождении в критическую секцию нескольких параллельных процессов, гарантию отсутствия бесконечного ожидания ресурса некоторым процессом, гарантию достижения некоторой цели (за заданное время) и т.д.

В дискретном темпоральном подходе предполагается, что любое свойство поведения вычислительной модели можно определить перечислением всех последовательностей состояний, которые обладают этим свойством. Другими словами, каждое свойство поведения в дискретном темпоральном подходе определяется некоторым множеством последовательностей $\Pi \subseteq \Sigma^{\infty}$. Множество Π называется финитным (финитное свойство), если оно содержит только конечные последовательности, и инфинитным, если оно содержит только бесконечные последовательности (инфинитное свойство).

Качественные требования к поведению могут быть классифицированы. Классификация требований к поведению определяется через классификацию возможных последовательностей состояний.

На линейных семантиках среди требований к поведению наиболее часто выделяются два основных класса (классификация предложена Л.Лэмпортом [59],[60])— требования *корректности* (*safety*, гарантия того, что "нечто" сохраняется во всех состояниях всех вычислений вычислительной модели) и требования *жизнеспособности* (*liveness*, гарантия того, что "нечто" случится когда-нибудь в каждом вычислении вычислительной модели). Соответственно на множестве последовательностей Σ^{∞} выделяются два подмножества Π_S и Π_L.

Назовем конечную последовательность $\sigma \in \Sigma^{+}$ финитным префиксом последовательности $\sigma' \in \Sigma^{\infty}$, если $\sigma = s_0, \ldots, s_k, \quad \sigma' = s_0, \ldots, s_k, \ldots$. Обозначим через $\sigma \prec \sigma'$ тот факт, что $\sigma \in \Sigma^{+}$ является финитным префиксом $\sigma' \in \Sigma^{\infty}$, но отличается от нее ($\sigma \neq \sigma'$). Будем писать $\sigma \preceq \sigma'$, если $\sigma \prec \sigma' \vee \ \sigma = \sigma' \in \Sigma^{+}$.

Множества бесконечных последовательностей $\Pi_S, \Pi_L \subseteq \Sigma^\omega$, соответствующие классам safety и liveness, строятся по следующему принципу:

Класс safety : $\sigma \in \Pi_S \Leftrightarrow \forall\sigma'(\sigma' \preceq \sigma) : \exists\sigma''(\sigma'' \in \Sigma^\omega) : \sigma' \cdot \sigma'' \in \Pi_S$

Класс liveness : $\sigma' \in \Pi_L \Rightarrow \forall\sigma(\sigma \in \Sigma^+) : \sigma \cdot \sigma' \in \Pi_L$

Последовательность σ попадет в множество Π_S тогда, когда все ее финитные префиксы могут быть достроены до бесконечной последовательности, которая попадет в Π_S. В множество Π_L попадут все бесконечные последовательности с произвольным началом, такие, что их конец попадает в Π_L.

Другая классификация свойств на линейных семантиках предложена в [63] (Борелл-классификация). В ней иерархически выстроены классы качественных требований к поведению вычислительной модели параллельной системы, которые могут быть формализованы в дискретном темпоральном подходе:

• *safety* – утверждение, что "нечто" "случается" во всех состояниях каждого вычисления из множества всевозможных вычислений

• *termination* – утверждение, что "нечто" "случается" хотя бы в одном состоянии каждого вычисления из множества вычислений

• *intermittence* – объединение классов *safety* и *termination* на множестве вычислений

• *recurrence* – утверждение, что "нечто" "случается" в бесконечно многих состояниях каждого вычисления из множества вычислений, включает в себя классы *safety, termination, intermittence*

• *persistence* – утверждение, что "нечто" "случается" во всех состояниях каждого вычисления из множества вычислений, начиная с некоторого состояния, включает в себя классы *safety, termination, intermittence*

• *progress* – объединение классов *persistence* и *recurrence*

Сопоставим базовому свойству (тому, что мы назвали "нечто") некоторое финитное множество Π. Для определения классов safety, termination, recurrence и persistence введем соответственно отображения A, E, R и P, описывающие правила конструирования соответствующих инфинитных множеств Π' из базового финитного множества Π:

$$A:\ \Pi \to \Pi' \qquad \sigma \in \Pi' = A(\Pi) \quad \Leftrightarrow \quad \forall \sigma'(\sigma' \preceq \sigma) : \sigma' \in \Pi$$
$$E:\ \Pi \to \Pi' \qquad \sigma \in \Pi' = E(\Pi) \quad \Leftrightarrow \quad \exists \sigma'(\sigma' \preceq \sigma) : \sigma' \in \Pi$$
$$R:\ \Pi \to \Pi' \qquad \sigma \in \Pi' = R(\Pi) \quad \Leftrightarrow$$
$$\forall \sigma'(\sigma' \preceq \sigma) : \exists \sigma''(\sigma' \preceq \sigma'' \preceq \sigma), \sigma'' \in \Pi$$
$$P:\ \Pi \to \Pi' \qquad \sigma \in \Pi' = P(\Pi) \quad \Leftrightarrow$$
$$\exists \sigma'(\sigma' \preceq \sigma) : \forall \sigma''(\sigma' \preceq \sigma'' \preceq \sigma), \sigma'' \in \Pi$$

Последовательность σ попадает в множество $A(\Pi)$ только если все его конечные префиксы содержались в множестве Π. Последовательность σ попадает в множество $E(\Pi)$ только если некоторый его конечный префикс содержался в множестве Π. Последовательность σ попадает в множество $R(\Pi)$ либо если она конечна и содержится в Π, либо если она бесконечна и бесконечно много ее финитных префиксов содержались в Π. Последовательность σ попадает в множество $P(\Pi)$ либо если она конечна и содержится в Π, либо если она бесконечна и у нее имеется некоторый финитный префикс, начиная с которого все ее финитные префиксы содержатся в Π.

В работе [63] показано, что множества последовательностей Π_s и $A(\Pi)$ эквивалентны и, таким образом, классы safety в Борелл-классификации и в классификации Л.Лэмпорта совпадают.

Конструирование множеств последовательностей для классов intermittence и progress происходит объединением вышеописанных множеств:

$$\Pi_I = A(\Pi') \cup E(\Pi'')$$
$$\Pi_P = R(\Pi') \cup P(\Pi'').$$

где Π', Π'' – базовые финитные множества.

На семантиках деревьев вычислений используется классификация Л.Лэмпорта, но в классе liveness (на множестве Π_L) выделяются дополнительно два подкласса: *неизбежность* (*inevitability*, гарантия того, что "нечто" случится когда-нибудь на каждой ветви дерева вычислений вычислительной модели) и *возможность* (*possibility*, гарантия того, что "нечто" случится когда-нибудь хотя бы на одной ветви дерева вычислений).

Для формального описания свойств поведения, которым должна удовлетворять модель, в дискретном темпоральном подходе разработан язык спецификации — темпоральная логика [63, 64].

В отличие от формул логики исчисления предикатов первого порядка, темпоральные формулы могут описывать не только истинность некоторого утверждения в конкретном состоянии вычисления, но и длительность истинности утверждения на последовательности состояний.

Логические формулы являются статическими, в каждый момент наблюдения за поведением модели их истинность не меняется. Истинность же темпоральных формул, напротив, зависит от момента наблюдения, то есть привязана к позиции вычисления модели.

Каждому типу семантики соответствует своя темпоральная логика.

Термами темпоральной логики являются предикаты, которые на отдельных состояниях вычисления принимают значения "истина" или "ложь". Ими могут являться формулы исчисления предикатов первого порядка, включающие стандартные операции отношений над целыми числами.

Темпоральная формула конструируется из термов с применением логических операторов $\neg$ и $\vee$ и так называемых темпоральных операторов $\bigcirc$("Next") и $\mathcal{U}$("Until").

Пусть имеется вычисление $\sigma : s_0, s_1, \ldots,$ каждому состоянию s_i соответствует множество значений переменных состояния. Если последовательность конечная, то есть $\sigma = s_0, \ldots, s_k$, то определим ее длину $|\sigma| = k + 1$, в противном случае будем писать $|\sigma| = \omega$.

Для заданного вычисления σ определим индуктивно понятие *истинности* темпоральной формулы p в состоянии s_i (в позиции i, $0 < i < |\sigma|$), вычисления. Истинность темпоральной формулы p в позиции i вычисления σ будем обозначать $(\sigma, i) \models p$.

Если p является логической формулой (термом), то

$$(\sigma, i) \models p \quad \Leftrightarrow \quad s_i \models p.$$

Истинность терма может быть логически вычислена в соответствии со значениями переменных состояния, участвующих в p, в состоянии s_i.

Пусть теперь q, p – термы.

$$
\begin{array}{lcl}
(\sigma, i) \models \neg p & \Leftrightarrow & (\sigma, i) \not\models p \\
(\sigma, i) \models p \vee q & \Leftrightarrow & (\sigma, i) \models p \quad \vee \quad (\sigma, i) \models q \\
(\sigma, i) \models \bigcirc q & \Leftrightarrow & i + 1 < |\sigma| \wedge (\sigma, i + 1) \models q \\
(\sigma, i) \models p \mathcal{U} q & \Leftrightarrow & \forall k (i \leq k < |\sigma|) : \\
 & & [(\sigma, k) \models p \vee \exists j (i \leq j \leq k) : (\sigma, j) \models q]
\end{array}
$$

Истинность темпоральной формулы $\bigcirc q$ в некоторой позиции i вычисления означает, что в следующем состоянии (если оно существует) будет истинен терм q. Таким образом, одноместный оператор "Next" позволяет описать истинность логической формулы (некоторого свойства поведения) непосредственно в следующем состоянии вычисления.

Истинность темпоральной формулы $p \mathcal{U} q$ в некоторой позиции i вычисления означает, что в последующих состояниях будет истинен терм p до тех пор, пока не найдется состояние, в котором будет истинен терм q (или до конца цепочки). Таким образом, двуместный оператор "Until" описывает длительность истинности одной логической формулы (некоторого свойства поведения), первого операнда, относительно некоторой другой логической формулы (другого свойства поведения), второго операнда.

На базе темпорального оператора "Until" для семантик деревьев вычислений в темпоральную логику введены два оператора "$\forall \mathcal{U}$" и "$\exists \mathcal{U}$", в которых используются два квантора — "$\exists$" ("для некоторого пути в дереве вычислений") и "$\forall$" ("для всех путей в дереве вычислений"), со следующей семантикой:

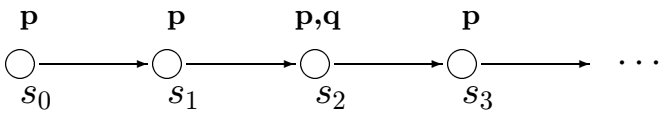
$f_1 \exists \mathcal{U} f_2$ ("$\exists$-*Until*") $\Leftrightarrow f_1 \mathcal{U} f_2$ — по нек. пути дерева вычислений

$f_1 \forall \mathcal{U} f_2$ ("$\forall$-*Until*") $\Leftrightarrow f_1 \mathcal{U} f_2$ — по всем путям дерева вычислений

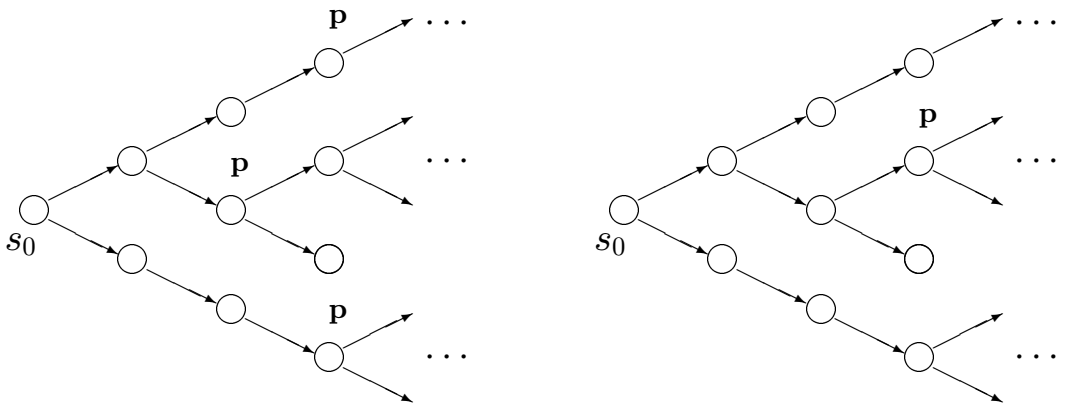
Теперь определим понятие *истинности* темпоральной формулы p над вычислением σ (над деревом вычислений для семантики деревьев вычислений), которое определяется как истинность темпоральной формулы в начальной позиции вычисления (в корне дерева вычислений для семантики деревьев вычислений). Истинность темпоральной формулы p над вычислением σ будем обозначать $\sigma \models p$:

$$\sigma \models p \;\Leftrightarrow\; (\sigma, 0) \models p.$$

Для описания основных классов требований к поведению в темпоральную логику введены также дополнительные темпоральные опе-



а) требования корректности: $\Box p$ и жизнеспособности $\Diamond q$



б) неизбежность: $\forall\Diamond p$ в) возможность: $\exists\Diamond p$

Рис. 2: Базовые требования к поведению.

раторы. Для линейных семантик:

$\Box$ (*"Всегда"*)	$\Box p = p\mathcal{U}False$	safety
$\Diamond$ (*"Когда-нибудь"*)	$\Diamond q = \neg\Box\neg q\mathcal{U}False$	liveness

Для семантик деревьев вычислений:

$\Box$ (*"Всегда"*)	$\Box p = p\mathcal{U}False$	safety
$\forall\Diamond$ (*"Когда-нибудь всюду"*)	$\forall\Diamond q = \neg\Box\neg q\forall\mathcal{U}False$	inevitability
$\exists\Diamond$ (*"Когда-нибудь где-то"*)	$\exists\Diamond q = \neg\Box\neg q\exists\mathcal{U}False$	possibility

На рисунке 2 изображены вышеописанные классы требований к поведению.

Темпоральная логика для линейных семантик называется линейной темпоральной логикой [63], темпоральная логика для семантик деревьев вычислений называется темпоральной логикой деревьев вычислений [37].

Требования к поведению по Борелл-классификации специфицируются следующими темпоральными формулами ($p, q \in$ TL):

safety – $\Box\ p$

termination – $\Diamond p, p\mathcal{U}q$

recurrence – $\Box\Diamond p, \Box(p \rightarrow \Diamond q)$
persistence – $\Diamond\Box p$
intermittence – $\Box p \vee \Diamond q$
progress – $\Box\Diamond p \vee \Diamond\Box q, \Box\Diamond p \Rightarrow \Box\Diamond q$

Таким образом, на языке темпоральной логики можно описать качественные требования к поведению модели.

Для спецификации временных требований к поведению используется расширение темпоральной логики, которое называется темпоральной логикой реального времени (Real Time temporal logic) [45, 62]. В темпоральной логике реального времени для всех темпоральных операторов определены конкретные временные границы, например:

$$\Diamond_{[l;u]}\ p$$

где $l, u \in R^{\geq}$ – нижняя и верхняя временные границы возможности истинности свойства поведения p ("когда-нибудь, но не раньше чем через l и не позже чем через u единиц времени").

Темпоральная логика реального времени используется для спецификации свойств поведения вычислительных моделей систем реального времени.

В настоящей работе нам понадобится также использовать темпоральную логику реального времени деревьев вычислений (Timed Computation Tree Logic) [42], разработанную для семантик деревьев вычислений, формулы которой имеют вид:

$$\delta\ ::=\ p\ \mid\ \neg\delta\ \mid\ \delta_1 \vee \delta_2\ \mid\ \delta_1 \exists\mathcal{U}_{\sharp n}\delta_2\ \mid\ \delta_1 \forall\mathcal{U}_{\sharp n}\delta_2$$

где p – терм, "$\sharp$" $\in \{\ <\ ,\ \leq\ ,\ =\ ,\ >\ ,\ \geq\ \}$, $n \in Q$.

Эта логика используется для спецификации свойств гибридных автоматов.

Для спецификации требований к поведению гибридных автоматов используются также расширения темпоральной логики реального времени (интервальная темпоральная логика [48], гибридная темпоральная логика [46]).

1.4 Метод символьной верификации и система HyTech

В данном параграфе представлен эвристический алгоритм построения характеристического множества гибридного автомата и разработанный на базе этого алгоритма метод символьной верификации гибридных автоматов [67].

Пусть дан гибридный автомат $H = \{X(D), V, E, \Sigma, flow, pre, post, inv, init, syn\}$.

Назовем *функцией эволюции* функцию вида $\varphi : [V, D \times R_{\geq 0}] \rightarrow D$, определяющую значение вектора переменных X через время t после входа в локацию v при начальных условиях X_0. Будем обозначать его $\varphi_v[X_0](t)$. Если локальные поведения $flow$ фазовых переменных гибридного автомата могут быть заданы произвольным образом (дифференциальными уравнениями, дифференциальными включениями, явными функциями, предикатами), то функция эволюции есть локальное поведение, записанное в виде функции и обладающее следующими свойствами:

для любых $v \in V, X_0 \in X, t \in R_{\geq 0}$,

1. $\varphi_v[X_0](0) = X_0$;
2. $\varphi_v[X_0](t + t') = \varphi_v[\varphi_v[X_0](t)](t')$.

Назовем *Предикатом возможности увеличения времени в локации* логическую функцию

$\texttt{tcp} : [V, D \times R_{\geq 0}] \rightarrow \{true, false\}$, определяющую максимальную длительность t нахождения в локации v при заданном начальном значении X_0. Будем обозначать ее $\texttt{tcp}_v[X_0](t)$. В качестве предикатов $\texttt{tcp}$ могут выступать логические формулы, обладающие свойством непрерывной истинности на протяжении всего времени нахождения в локации:

для любых $v \in V, X_0 \in X, t \in R_{\geq 0}$,

1. $\texttt{tcp}_v[X_0](0) = true$;
2. $\texttt{tcp}_v[X_0](t) \Rightarrow \forall t' \leq t \ \texttt{tcp}_v[X_0](t')$

Предикат возможности увеличения локального времени определяется различными способами[5].

[5]Например,он может быть определен через предикат-инвариант автомата и его функцию

Ставится задача вычисления характеристического множества для заданного гибридного автомата.

Предлагается нетрадиционный способ решения данной задачи [67], основанный на предположении, что множества достижимости локаций гибридного автомата представимы в виде логических формул над его фазовыми переменными. Такое предположение дает право исследовать пучки траекторий фазовых переменных, отвечающих различным пробегам гибридного автомата и объединять каждое множество пробегов, отвечающих одной и той же последовательности локаций, в так называемый символьный пробег гибридного автомата. Все поведение гибридного автомата при таком подходе будет описываться множеством его символьных пробегов.

Итак, пусть гибридный автомат имеет m локаций. Представим множество его начальных состояний, описанных предикатом $init$, в виде логической формулы $I = \bigvee_{i=1}^{m} I_i$, где терм I_i описывает множество начальных значений локации v_i (для начальных локаций, очевидно, $I = init(v_k)$, для всех других, $I \equiv false$).

Рассмотрим текущую локацию v и предположим, что множество ее начальных значений $\{X_0\}$ является областью фазового пространства,

---

эволюции:

$$\texttt{tcp}_v[X_0](t) \;\equiv\; t = 0 \quad \vee \forall t' \leq t \;\; inv(v)[\varphi_v[X_0](t')]$$

Часто tcp-предикат определяется в соответствии с политикой синхронизации параллельных компонент в гибридном автомате. Рассмотрим основные случаи.

Пусть $P = \bigvee_{e=(v,v')\in E} pre(e)$ ($pre(e)$ – предикаты на дугах автомата, выходящих из локации v) и пусть $\varphi_v[X_0](t)$ – значение функции локального поведения в локации v в момент t при начальных условиях X_0. Варианты предиката $\texttt{tcp}_v[X](t)$:

$$\texttt{tcp}_v[X_0](t) \equiv \exists t' \geq t \;\; P(\varphi_v[X_0](t')) \vee (\forall t' \neg P(\varphi_v[X_0](t'))) \qquad (***)$$

— истинен максимально возможное время пребывания в локации v,"пока хотя бы один переход еще возможен" (асинхронная политика)

$$\texttt{tcp}_v[X_0](t) \equiv \forall t' \leq t \;\; \neg P(\varphi_v[X_0](t'))$$

— истинен минимально возможное время пребывания в локации v, "до первого возможного перехода" (синхронизирующая политика)

$$\texttt{tcp}_v[X_0](t) \equiv \bigwedge_e \forall t' \leq t \;\; pre(e)[\varphi_v[X_0](t')] \vee (\forall t'' \leq t' \neg pre(e)[\varphi_v[X_0](t'')])$$

— истинен до первого отказа какого-нибудь перехода из локации v, "пока все еще возможны" (intermediate политика)

Возможность различных способов описания предиката tcp обеспечивает универсальность представленного далее метода символьной верификации для параллельных систем с различной политикой синхронизации.

которое может быть описано предикатом P.

Найти множество достижимости текущей локации в нашем подходе означает описать в виде логических формул преобразования фазовых переменных гибридного автомата на непрерывной и дискретной фазе локации.

Преобразование значений переменных на непрерывной фазе может быть описано логической формулой, которую будем обозначать "$\overrightarrow{P}^{v}$":

$$\overrightarrow{P}^{v}(X) \equiv \exists X_0 \; \exists t \; P(X_0) \wedge \mathtt{tcp}_v[X_0](t) \wedge X = \varphi_v[X_0](t)$$

Формально формула $\overrightarrow{P}^{v}$ является конъюнкцией предиката $P(X_0)$, описывающего область начальных значений локации v, tcp-предиката этой локации и функции эволюции вектора фазовых переменных. С помощью формальных символьных преобразований из формулы $\overrightarrow{P}^{v}$ удаляются кванторы существования с подстановкой переменной t из tcp-предиката в функцию эволюции и, таким образом, предикат $\overrightarrow{P}^{v}$ описывает множество начальных значений переменных $\{X_0\}$ в v и преобразование этого множества на момент выхода из локации v. Будем говорить, что формула $\overrightarrow{P}^{v}$ описывает эволюцию формулы P в локации v.

Пусть теперь P – предикат, описывающий множество значений фазовых переменных после непрерывной фазы в локации v (то есть на выходе из локации v), и пусть e – некоторая дуга, выходящая из этой локации. Преобразование значений фазовых переменных на дискретной фазе поведения гибридного автомата по дуге e может быть описано логической формулой, которую будем обозначать "$\mathtt{post}_e[P]$":

$$\mathtt{post}_e[P](X) \equiv \exists X_0 \; P(X_0) \wedge pre_e(X_0) \wedge X = post_e(X_0)$$

Формула $\mathtt{post}_e[P]$ является конъюнкцией предиката $P(X_0)$, описывающего область значений вектора переменных $\{X_0\}$ непосредственно перед переходом по дуге e, предиката, описывающего ограничения на значения переменных, при которых переход по дуге возможен, и формулу преобразования переменных на дуге. С помощью формальных символьных преобразований из формулы удаляется квантор существования и, таким образом, предикат $\mathtt{post}_e[P]$ описывает множество значений переменных $\{X_0\}$ перед переходом по дуге e и преобразование этого множества после перехода. Будем говорить, что фор-

мула $\texttt{post}_e[P]$ описывает эволюцию формулы P при переходе по дуге e.

Введем понятие *символьного пробега* как следующую последовательность пар:

$$(v_0, P_0)(v_1, P_1)\dots(v_i, P_i)\dots$$

$P_0 \equiv I_0$ и для $\forall i \geq 0$ существует дуга $e_i : v_i \to v_{i+1}$ и $P_{i+1} = \texttt{post}_{e_i}[\overrightarrow{P_i}^{v_i}]$.

Совокупность всевозможных символьных пробегов гибридного автомата назовем символьным поведением гибридного автомата.

Будем теперь строить последовательно множества достижимых состояний гибридного автомата из произвольной локации v_0 и начального множества состояний, описанных предикатом I_0. На i-том шаге итерации, очевидно, множество достижимых состояний гибридного автомата будет описываться в виде дизъюнкции предикатов $P_i = \bigvee_{k=0}^{i} P_k$ его всевозможных символьных пробегов, начинающихся из v_0. Если на каком-то шаге итерации i окажется, что $P_{i+1} \equiv P_i$, то будем говорить, что наш итерационный процесс сходится и назовем предикат P_i стационарной точкой итерационного процесса. Выход итерационного процесса на стационарную точку будет означать, что предикат P_i описывает множество всех состояний автомата, достижимых из локации v_0 и начального множества состояний I_0.

Применив данный итерационный алгоритм для всех локаций автомата, мы получим (в случае его сходимости) некоторый предикат P, который будем называть характеристическим предикатом гибридного автомата.

Алгоритм построения характеристического предиката. *Характеристический предикат P гибридного автомата H есть дизъюнкция $\bigvee_{i=1}^{m} P_i$, где каждый P_i соответствует локации v_i и является первой найденной стационарной точкой следующего итерационного процесса:*

а) $i = 0 : P_k^{(0)} \equiv \overrightarrow{I_k}^{v_k}, \quad 1 \leq k \leq m;$

б) дано: $P_k^{(i)}, \quad 1 \leq k \leq m,$

итерационная формула для $P_k^{(i+1)}$*:*

$$P_k^{(i+1)} = P_k^{(i)} \vee \bigvee_j \overrightarrow{\texttt{post}_e[P_j^{(i)}]}^{v_k} \tag{5}$$

где $1 \leq k \leq m, e = (v_j, v_k)$ Обозначим через [Q] множество значений фазовых переменных, заданных некоторым предикатом Q. В работе [33] доказывается следующее важное утверждение о характеристическом предикате:

Теорема 1.4.1 *В случае сходимости итерационного алгоритма для гибридного автомата H с множеством начальных значений, заданных логической формулой I, множество значений фазовых переменных, которое задается построенным характеристическим предикатом P, равно характеристическому множеству этого гибридного автомата:*

$$[P] = Reach(H).$$

Теорема 1.4.1 является обоснованием возможности использования символьного подхода для построения характеристического множества гибридного автомата.

Рассмотрим теперь обратную задачу.

Пусть дан гибридный автомат $H = \{X(D), V, E, \Sigma, flow, pre, post, inv, init, syn\}$. Пусть задан предикат R, описывающий некоторую область фазового пространства гибридного автомата.

Ставится задача найти множество всех предшествующих состояний гибридного автомата, из которых достижима заданная область фазового пространства. Обозначим это множество через $Pre(H, R)$. Будем решать эту задачу аналогичным образом.

Предположим, что предикат R может быть представлен в виде дизъюнкции $R = \bigvee_{i=1}^{m} R_i$, где m – количество локаций автомата. Например, если исследуется достижимость локации i, то $R_i \equiv (l = i), R_{j \neq i} \equiv false$, где l – дополнительная переменная, описывающая номер текущей локации.

Предикат R назовем целевым предикатом.

Введем две логические формулы, описывающие множества возможных начальных значений фазовых переменных для дискретных и непрерывных фаз поведения гибридного автомата в текущей локации v.

Предположим, что множество текущих значений фазовых переменных $\{X_{cur}\}$ в локации v гибридного автомата может быть описано предикатом P. Тогда множество значений $\{X\}$, преобразование которых на непрерывной фазе поведения гибридного автомата даст текущее значение X_{cur}, может быть описано логической формулой, которую будем обозначать "$\overleftarrow{P}^{v}$":

$$\overleftarrow{P}^{v}(X) = \exists X_{cur}\ \exists t \quad P(X_{cur}) \wedge \mathtt{tcp}_v(X)(t) \wedge X_{cur} = \varphi_v(X)(t) \equiv$$
$$\equiv \exists t \quad \mathtt{tcp}_v(X)(t) \wedge P(\varphi_v(X)(t))$$

Как и в предыдущем случае, из формулы $\overleftarrow{P}^{v}$ удаляются кванторы существования с подстановкой переменной t из tcp-предиката в функцию эволюции и, таким образом, предикат $\overleftarrow{P}^{v}$ описывает множество значений переменных $\{X\}$, которые в результате эволюции в локации v превращаются в множество текущих значений $\{X_{cur}\}$, удовлетворяющих предикату P (то есть их которых достижимы значения $\{X_{cur}\}$). Будем говорить, что формула $\overrightarrow{P}^{v}$ описывает предысторию формулы P в локации v.

Пусть теперь P – предикат, описывающий множество начальных значений фазовых переменных $\{X_0\}$ в текущей локации v и пусть e – некоторая дуга, входящая в эту локацию. Множество значений переменных, преобразование которых на дискретной фазе поведения гибридного автомата по дуге e, даст текущее значение X, может быть описано логической формулой, которое будем обозначать "$\mathtt{pre}_e[P]$":

$$\mathtt{pre}_e[P](X) = \exists X_0 P(X_0) \wedge pre_e(X) \wedge X_0 = post_e(X) \equiv$$

$$\equiv pre_e(X) \wedge P(post_e(X))$$

Формула $\mathtt{pre}_e[P]$ описывает множество значений переменных $\{X\}$, которые в результате перехода по дуге e превращаются в множество начальных значений $\{X_0\}$ локации v, удовлетворяющих предикату P (то есть их которых достижимы значения $\{X_0\}$). Будем говорить, что формула $\mathtt{pre}_e[P]$ описывает предысторию формулы P на переходе по дуге e.

Будем строить, как и в случае построения характеристического предиката, последовательно множества состояний гибридного автомата, из которых достижимы текущие состояния, для каждой лока-

ции v_i, начиная из множества состояний, описанных предикатом R_i.

Применив данный итерационный алгоритм для всех локаций автомата, мы получим (в случае его сходимости) некоторый предикат, который назовем предикатом предшествующих состояний.

Алгоритм построения предиката предшествующих состояний целевого предиката. *Для множества состояний гибридного автомата, описанных некоторым предикатом* $R = \bigvee_{i=1}^{n} R_i$, *(*$R_i$ *— целевой предикат локации* v_i*), предикат предшествующих состояний* P *является дизъюнкцией* $\bigvee_{i=1}^{m} P_i$, *где каждый* P_i *соответствует локации* v_i *и является первой найденной стационарной точкой следующего итерационного процесса:*

а) $i = 0 : P_k^{(0)} \equiv \overleftarrow{R_k}^{v_k}$, $1 \le k \le m$;

б) дано: $P_k^{(i)}$, $1 \le k \le m$,

итерационная формула для $P_k^{(i+1)}$*:*

$$P_k^{(i+1)} = P_k^{(i)} \vee \bigvee_j \overleftarrow{\texttt{pre}_e[P_j^{(i)}]}^{v_k} \tag{6}$$

где $1 \le k \le m, e = (v_j, v_k)$ В работе [33] доказывается следующее утверждение о предикате предшествующих состояний:

Теорема 1.4.2 *В случае сходимости итерационного алгоритма для гибридного автомата* H *с множеством состояний, заданных целевой логической формулой* R, *множество состояний, которое задается построенным предикатом предшествующих состояний* P, *равно множеству всех предшествующих состояний гибридного автомата, из которых достижимо целевое множество состояний:*

$$[P] = Pre(H, R).$$

Теорема 1.4.2 является обоснованием возможности использования символьного подхода для построения множества всех предшествующих состояний гибридного автомата.

Рассмотрим теперь в качестве начального предиката гибридного автомата I (или целевого предиката R) темпоральную формулу (будем рассматривать темпоральную логику реального времени деревьев вычислений) и поставим задачу построения характеристического множества данного гибридного автомата. Темпоральная формула

будет описывать дополнительные темпоральные ограничения на множество пробегов автомата.

Предлагается следующий эвристический алгоритм построения характеристического множества (или множества предшествующих состояний) гибридного автомата с множеством начальных состояний (или с целевым предикатом), описанным темпоральной формулой [67]:

вначале находятся характеристические предикаты (или предикаты предшествующих состояний) для гибридного автомата H с начальными (или целевыми) предикатами, являющимися термами исходной темпоральной формулы;

далее для каждого класса темпоральных свойств применяется свое итерационное правило построения характеристического предиката.

Заметим, что темпоральные операторы $\Box$, $\forall\mathcal{U}$ и $\exists\mathcal{U}$, используемые для описания основных классов требований к поведению, выражаются через базовый темпоральный оператор $\mathcal{U}$("Until") (см.п. 1.3). Таким образом, все итерационные правила строятся на базе итерационного правила построения характеристического предиката для темпоральной формулы $p\mathcal{U}q$.

Введем бинарный оператор "$\triangleright$" ("single-step until"), который описывает преобразование множества значений фазовых переменных гибридного автомата, удовлетворяющих темпоральной формуле $p\mathcal{U}q$, на одном шаге итерационного процесса, другими словами, можно сказать, что этот оператор описывает, как модифицируется формула $p\mathcal{U}q$ при переходе от текущего состояния гибридного автомата к следующему. Этот оператор может быть определен как через предысторию текущего множество состояний, так и через множество состояний, достижимых из текущего состояния. Определим "$\triangleright$" через предысторию.

Пусть P P' — две логические формулы над X. Предикат $P' \triangleright P$ описывает множество состояний $\{X'\}$, из которого можно попасть в множество $\{X\}$, на котором истинен предикат P, в результате одного шага по пробегу гибридного автомата так, чтобы в течение всей эволюции множества $\{X'\}$ был бы истинен предикат $P' \vee P$. Формально:

$$(P' \triangleright P)(X) \equiv \exists e, \exists t(\mathtt{pre}_e[P] \vee P)(\varphi_v[X](t)) \wedge$$
$$(\mathtt{tcp}_v[X](t) \wedge \forall t' \leq t(P' \vee P)(\varphi_v[X](t'))$$

Таким образом, множество состояний гибридного автомата, удовлетворяющих темпоральной формуле $P'\mathcal{U}P$, можно вычислить итерационно через множества состояний, удовлетворяющих логическим формулам P и P', используя оператор $\triangleright$. Сам же характеристический предикат для темпоральной формулы вида $p\mathcal{U}q$ будет являться (не темпоральной) логической формулой.

Пусть ϕ_1, ϕ_2 и ϕ – термы темпоральных формул, а $P_{\phi_1}, P_{\phi_2}, P_{\phi}$ – характеристические предикаты гибридного автомата H с начальными (или целевыми) предикатами ϕ_1, ϕ_2 и ϕ соответственно. Пусть "$\sharp$" $= \{<, \leq, =, \geq, >\}, n \in Q$, а t – дополнительная переменная-таймер гибридного автомата, отвечающая за системное время в гибридном автомате (во всех локациях $\dot{t} = 1$).

Правило 1. Характеристический предикат формулы $\phi_1 \exists\mathcal{U}_{\sharp n}\phi_2$ есть $\vee_i P_i(0)$, где:

- $P_0(t) \equiv t \sharp n \wedge P_{\phi_2}$
- $\forall i \geq 0,\ P_{i+1}(t) = P_i(t) \vee P_{\phi_1} \triangleright P_i(t)$

Правило 2. Характеристический предикат формулы $\Box\phi$ есть $\wedge_i P_i$, где:

- $P_0 \equiv P_{\phi}$
- $\forall i \geq 0,\ P_{i+1} = P_i \wedge \neg(\text{true} \triangleright \neg P_i)$

Правило 3. Характеристический предикат формулы $\forall\Diamond_{\leq n}\phi$ есть $\neg \vee_i P_i(0)$, где:

- $P_0(t) \equiv t > n$
- $\forall i \geq 0,\ P_{i+1}(t) = P_i(t) \vee \neg P_{\phi} \triangleright P_i(t)$

Для каждого из итерационных правил, в случае их сходимости, в работе [33] представлено доказательство утверждения о равенстве множества состояний, описанного построенным характеристическим предикатом, и характеристическим множеством гибридного автомата с начальным множеством состояний, описанных в виде соответствующей темпоральной формулы. Доказательство является обоснованием возможности использования символьного подхода для построения характеристического множества гибридного автомата с темпоральными формулами в качестве начальных предикатов.

Наконец, рассмотрим задачу верификации гибридного автомата.

Опишем характеристики отдельных локаций (ограничения на длительность нахождения в каждой локации, области возможных значений фазовых переменных в локациях, наступление событий из алфавита событий и пр.) в виде обычных логических формул и припишем их к соответствующим локациям.

Специфицируем требования к поведению гибридного автомата, представленной семантикой деревьев вычислений, темпоральными формулами реального времени деревьев вычислений (см. 1.3), в которых в качестве термов могут участвовать логические формулы вида:

$$p \quad = \quad l \leq x_i \leq u \quad | \quad a \quad | \quad \sum_i h_i x_i \sim c \quad | \quad q$$

где $x_i \in X$ – фазовые переменные гибридного автомата, $a \in \Sigma$ – метка алфавита событий, q – абстрактный логический терм, $l, u, h_i, c \in R$, $\sim = \{<, \leq, =, \geq, >\}$.

Задача верификации формулируется следующим образом: ответить на вопрос, отвечает ли поведение заданного гибридного автомата $H = \{X(D), V, E, \Sigma, flow, pre, post, inv, init, syn\}$ некоторому свойству, специфицированному в виде темпоральной формулы φ.

Заметим, что истинность темпоральной формулы определяется как ее истинность в начальном состоянии некоторого вычисления (в корне дерева вычислений в нашем случае). Поэтому можно решать задачу верификации, используя алгоритм построения характеристического предиката.

Заменим предикат начальных состояний гибридного автомата *init* на темпоральную формулу φ и будем искать характеристическое множество для такого гибридного автомата. Построенный характеристический предикат будет описывать характеристическое множество гибридного автомата, удовлетворяющее соответствующему требованию к поведению. Если окажется, что множество $[\varphi]$ не включает в себя множество состояний, достижимых из множества [init] начальных состояний гибридного автомата, то гибридный автомат не будет удовлетворять соответствующему требованию к поведению.

Задача верификации может быть решена и другим способом, используя алгоритм построения предшествующих состояний.

Найдем предикат, описывающий множество всех предшествующих состояний, из которых достижимы состояния, удовлетворяющие целевому темпоральному предикату φ. Если это множество не включает в себя множество начальных состояний гибридного автомата, то гибридный автомат не будет удовлетворять соответствующему требованию к поведению.

Данный подход к верификации был назван методом символьной верификации.

Основная трудность программной реализации описанных итерационных алгоритмов заключается в высокой вычислительной сложности реализации основных операций над логическими формулами, в частности проверки эквивалентности двух логических формул. Несмотря на то, что сами алгоритмы были разработаны сравнительно давно [43], построение автоматических систем символьной верификации стало возможно только после изобретения бинарных диаграмм (Binary Decision Diagramms, BDD's) для эффективного представления в памяти ЭВМ логических формул. Использование BDD's в методе символьной верификации обеспечивает линейное время выполнения основных операций над логическими формулами и позволяет эффективно представлять сам гибридный автомат в символьном виде.

В настоящей работе мы не будем уделять внимание реализационным аспектам метода символьной верификации, подробнее с BDD's можно ознакомиться в [40, 39].

Метод символьной верификации лег в основу системы автоматической верификации гибридных систем HyTech, разработанной в 1993 году группой, возглавляемой Т.Хензингером и П.-Т.Хо. Первая версия системы (1993г. [48],[33]) была реализована на базе пакета символьных вычислений Mathematica [74]. Последняя (третья) версия системы (1995–1997гг.) базируется на объектно–ориентированном подходе и целиком реализована средствами C++ [51]. Использование объектно–ориентированного подхода позволило расширить возможности манипуляции областями фазового пространства, использования синхронизации и приоритетов событий при верификации параллельных композиций гибридных автоматов.

1.5 Два примера символьной верификации

Рассмотрим два примера символьной верификации систем автоматического управления.

Пример 1. Система контроля газовой безопасности (прямой метод символьной верификации)

Система Gas Burner, используемая в качестве примера во многих работах гибридного направления [33, 67], представляет из себя тривиальную "игрушечную" с точки зрения теории управления модель динамической системы, которая, однако, оказывается удобной для "ручной" демонстрации различных алгоритмов верификации (в частности, прямого метода символьной верификации). Пример примечателен тем, что для него итерационный алгоритм построения характеристического предиката не сходится, однако дает итерационную формулу для всех достижимых состояний, с помощью которой можно доказать необходимое требование.

Управляемая система состоит из газового хранилища (танкера), кнопки запроса на пользование газом и системы контроля, которая может при необходимости прекратить или не разрешить очередной сеанс пользования газом (при помощи клапана). Система имеет два основные режима работы: "leak" (клапан открыт) и "$\neg$ leak" (клапан закрыт). При открытом клапане некоторое количество газа утекает, причем скорость утечки неизвестна, поэтому единственным способом контролировать процесс утечки является контроль за временем пользования газом.

Целью системы контроля является недопущение критической концентрации газа в окружающей среде. Для этого вводятся два временных ограничения на сеансы пользования газом:

(*) каждый сеанс не должен продолжаться дольше n единицы времени;

(**) интервал времени между двумя последовательными сеансами должен быть не менее m единиц времени.

Требование о недопустимости критической концентрации газа может быть сформулировано как требование к поведению системы Gas Burner следующим образом (требование корректности): *"Суммарная*

длительность сеансов пользования газом для интервалов наблюдения больших некоторого времени (k единиц времени) не должна превышать 5 процентов общего времени, независимо от начала наблюдения."

Верифицируем систему при конкретных значениях параметров n, m, k. Положим $n = 1, m = 30, k = 60$.

Верификация состоит из следующих шагов:

• Построение гибридного автомата для системы Gas Burner. Каждому режиму работы системы поставим в соответствие локацию гибридного автомата (локация 1 – режим "leak", локация 2 – режим "$\neg$ leak"). Введем в автомат три вещественные переменные, отслеживающие локальное время нахождения в каждой локации (переменная x), суммарную длительность сеансов пользования газом (переменная y) и суммарное время между сеансами (переменная z). Локальные поведения переменных опишутся следующим очевидным образом:

в локации 1 : $\dot{x} = 1, \dot{y} = 1, \dot{z} = 0$;

в локации 2 : $\dot{x} = 1, \dot{y} = 0, \dot{z} = 1$.

Переходы из одной локации в другую должны осуществляться под воздействием ограничений на сеансы пользования газом (*) и (**), описанных выше:

условие перехода $1 \to 2$: $\quad x \leq n$,

условие перехода $2 \to 1$: $\quad x \geq m$.

Значения некоторых переменных на переходах гибридного автомата могут изменяться, в нашем случае должна обнуляться на переходах переменная x.

В качестве начальной локации можно выбрать любую из двух, для определенности выберем локацию 1 и определим вектор начальных значений переменных гибридного автомата $x = 0, y = 0, z = 0$. Построенный гибридный автомат изображен на рис.3.

• Вычисление для каждой локации `tcp`-предиката по одной из предложенных формул (например, для асинхронной политики, см.формулу (***), сноска 5):

$$\texttt{tcp}_1[X_0](t) \equiv \exists t' \geq t \;\; x_0 + t \leq 1 \vee x_0 > 1 \quad \equiv \quad x_0 + t \leq 1$$

$$\texttt{tcp}_2[X_0](t) \equiv \exists t' \geq t \;\; x_0 + t \geq 30 \quad \equiv \quad true$$

Здесь и далее введены обозначения x_0, y_0, z_0 для начальных значе-

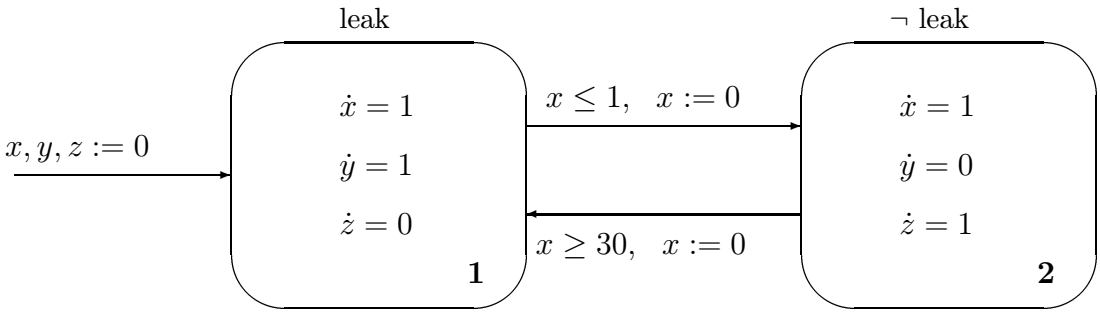


Рис. 3: Гибридный автомат системы контроля газовой безопасности.

ний переменных x, y, z в каждой локации.

- Описание требования корректности в виде логической формулы:

$$Pr \quad : \qquad y + z > 60 \Rightarrow 20y \le y + z \tag{7}$$

- Построение характеристического предиката гибридного автомата системы $P = P^1 \vee P^2$ по итерационным формулам:

база: начальный предикат $I = I^1 \vee I^2$, где $I^1 \equiv x = 0 \wedge y = 0 \wedge z = 0,\ I^2 \equiv false$

$$P_i^1 \ = \ P_{i-1}^1 \vee \overrightarrow{\qquad}^1_{\mathtt{post}_{21}[P_{i-1}^2]} \qquad P_i^2 \ = \ P_{i-1}^2 \vee \overrightarrow{\qquad}^2_{\mathtt{post}_{12}[P_{i-1}^1]} \tag{8}$$

в соответствии с (5) (P_i^1 описывает состояния, достижимые из локации 1 на i-том шаге итерационного процесса, P_i^2 — из локации 2):

$P_0^1 = \overrightarrow{I^1}^1 \equiv \exists x_0, y_0, z_0, \exists t$

$(x_0 = y_0 = z_0 = 0 \wedge x_0 + t \le 1 \wedge x = x_0 + t \wedge y = y_0 + t) =$

$z = 0 \wedge x = t \wedge y = x \wedge t \le 1 = z = 0 \wedge x \le 1 \wedge y - x = 0 \wedge y \le 1$

$P_0^2 \equiv false$

$$P_1^1 \ = \ P_0^1 \vee \overrightarrow{\qquad}^1_{\mathtt{post}_{21} false} \ = P_0^1$$

$P_1^2 \ = \ false \vee \overrightarrow{\qquad}^2_{\mathtt{post}_{12} P_0^1} \equiv \overrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^2_{\exists x_0, y_0, z_0 (z_0 = 0 \wedge y_0 - x_0 = 0 \wedge x_0 \le 1 \wedge x = 0)}$

$= \overrightarrow{\qquad\qquad\qquad\qquad}^2_{z_0 = 0 \wedge y_0 \le 1 \wedge x = 0} = \exists x_0, y_0, z_0, t \ (z_0 = 0 \wedge y_0 \le 1 \wedge x_0 = 0 \wedge x = x_0 + t \wedge z = z_0 + t \wedge y = y_0) = z - x = 0 \wedge y \le 1$

$P_2^2 \ = \ P_1^2 \vee \overrightarrow{\qquad}^2_{\mathtt{post}_{12} P_1^1} \ = P_1^2 \vee \overrightarrow{\qquad}^2_{\mathtt{post}_{12} P_0^1} \ = P_1^2$

$P_2^1 = P_1^1 \vee \overrightarrow{\texttt{post}_{21} P_1^2}^1 \equiv (z = 0 \wedge x \leq 1 \wedge y - x = 0 \wedge y \leq 1) \vee$
$\overrightarrow{\exists x_0, y_0, z_0 (z_0 - x_0 = 0 \wedge y_0 \leq 1 \wedge x_0 \geq 30 \wedge x = 0)}^1 = (z = 0 \wedge x \leq 1 \wedge y -$
$x = 0 \wedge y \leq 1) \vee \overrightarrow{z_0 \geq 30 \wedge y_0 \leq 1 \wedge x = 0}^1 = \exists x_0, y_0, z_0, t (z_0 \geq 30 \wedge y_0 \leq$
$1 \wedge x_0 = 0 \wedge x_0 + t \leq 1 \wedge x = x_0 + t \wedge z = z_0 \wedge y = y_0 + t) = \quad [\, z = 0 \wedge x \leq$
$1 \wedge y - x = 0 \wedge y \leq 1] \vee [x \leq 1 \wedge z \geq 30 \wedge y - x \leq 2\,]$

$P_3^1 = P_2^1 \vee \overrightarrow{\texttt{post}_{21} P_2^2}^2 = P_2^1 \vee \overrightarrow{\texttt{post}_{21} P_1^2}^2 = P_2^1$

$P_3^2 = P_2^2 \vee \overrightarrow{\texttt{post}_{12} P_2^1}^2 \equiv (z - x = 0 \wedge y \leq 1) \vee$
$\overrightarrow{\exists x_0, y_0, z_0 ([z_0 = 0 \wedge x_0 \leq 1 \wedge y_0 - x_0 = 0 \wedge y_0 \leq 1] \vee}$
$\overrightarrow{[x_0 \leq 1 \wedge z_0 \geq 30 \wedge y_0 - x_0 \leq 1] \wedge x_0 \leq 1 \wedge x = 0}^2 = (z - x = 0 \wedge y \leq 1) \vee$
$\overrightarrow{[z_0 = 0 \wedge x = 0 \wedge y_0 - x_0 = 0 \wedge x_0 \leq 1 \wedge y_0 \leq 1] \vee}$
$\overrightarrow{[x_0 \leq 1 \wedge z_0 \geq 30 \wedge y_0 - x_0 \leq 1 \wedge x = 0]}^1 = (z - x = 0 \wedge y \leq 1) \vee$
$\exists x_0, y_0, z_0, t ([z_0 = 0 \wedge x_0 = 0 \wedge y_0 - x_0 = 0 \wedge x_0 \leq 1 \wedge y_0 \leq 1] \vee$
$[x_0 \leq 1 \wedge z_0 \geq 30 \wedge y_0 - x_0 \leq 1 \wedge x_0 = 0]) \wedge (x = x_0 + t \wedge z = z_0 + t \wedge y = y_0) =$
$(z - x = 0 \wedge y \leq 1) \vee (z - x \geq 30 \wedge y \leq 2)$

Таким образом, получаем итерационную формулу для достижимых состояний:

$$P_n^1 = \begin{cases} P_{n-1}^1 & \text{n – нечетное,} \\ P_{n-1}^1 \vee x \leq 1 \wedge y - x \leq n \wedge z \geq 30n & \text{n – четное.} \end{cases}$$

$$P_n^2 = \begin{cases} P_{n-1}^2 & \text{n – четное,} \\ P_{n-1}^2 \vee y \leq n - 1 \wedge z - x \geq 30(n-2) & \text{n – нечетное.} \end{cases}$$

Общий вид характеристического предиката, содержащего все достижимые состояния гибридного автомата:

$$S \equiv [x \leq 1 \wedge y = x \wedge z = 0 \vee \exists n \geq 1 (x \leq 1 \wedge y - x \leq n \wedge 30n \leq z)]$$
$$\vee$$
$$[y \leq 1 \wedge z = x \vee \exists n \geq 1 (y \leq n + 1 \wedge 30n \leq z - x)]$$

• Верификация требования к поведению. Требование корректности будет выполняться на гибридном автомате, если предикат $\neg Pr$ не достижим из множества начальных предикатов. То есть, если S – характеристический предикат гибридного автомата, то верификация требования корректности сводится к доказательству, что $S \wedge \neg Pr = false$.

Так как $S \wedge \neg(y + z > 60 \Rightarrow 20y \leq y + z) = false$, то система Gas Burner удовлетворяет требованию корректности.

Пример 2. Система температурного контроля атомного реактора (обратный метод символьной верификации)

Этот демонстрационный пример рассмотрен в работах [33, 67]. Система температурного контроля атомного реактора состоит из датчика температуры в котле реактора и двух независимых подвижных стержней, с помощью которых реактор может охлаждаться. Целью системы контроля является поддержка температуры в реакторе в пределах между θ_m и θ_M. Самопроизвольно температура в реакторе увеличивается. При достижении верхнего критического значения θ_M, котел должен быть охлажден одним из стержней до нижнего критического значения θ_m. При этом использование каждого стержня возможно не чаще чем через T единиц времени после окончания его последнего использования. Если критическая температура θ_M в котле достигается раньше чем возможно использование обоих стержней, то необходима полная остановка реактора.

Параметр T, являющийся характеристикой подвижных стержней реактора, должен быть подобран таким образом, чтобы реактор не останавливался. В этом заключается основное требование к системе температурного контроля.

Для "ручной" демонстрации символьной верификации системы предположим, что температура в реакторе меняется линейно. Пусть v_r — скорость самопроизвольного увеличения температуры в реакторе, v_1 и v_2 — скорости охлаждения реактора с помощью первого и второго стержней соответственно $(v_r, v_1, v_2 > 0)$.

Верифицируем систему при конкретных значениях параметров $\theta_M, \theta_m, v_r, v_1, v_2, T$. Положим $\theta_M = 15, \theta_m = 3, v_r = 6, v_1 = 4, v_2 = 3, T = 6$.

Метод состоит из следующих шагов:

• Построение гибридного автомата (рис.4). Каждому режиму работы системы поставим в соответствие локацию гибридного автомата (локация v_0 – самопроизвольное увеличение температуры в реакторе при отсутствии стержней, локация v_1 – охлаждение реактора под

воздействием первого стержня, локация v_2 – охлаждение реактора под воздействием второго стержня, локация v_3 – полная остановка реактора). Множество переменных автомата будет включать в себя вещественную переменную θ для описания температуры реактора, две вещественные переменные x_1 и x_2, отслеживающие время между двумя последовательными использованиями каждого из стержней , а также дополнительную дискретную управляющую переменную l, которая принимает значение номера текущей локации.

Локальные поведения переменных опишутся следующим образом:
в локации $v_0 : \dot{\theta} = v_r, \dot{x}_1 = 1, \dot{x}_2 = 1;$
в локации $v_1 : \dot{\theta} = -v_1, \dot{x}_1 = 1, \dot{x}_2 = 1;$
в локации $v_2 : \dot{\theta} = -v_2, \dot{x}_1 = 1, \dot{x}_2 = 1;$
в локации $v_3 : \dot{\theta} = 0, \dot{x}_1 = 0, \dot{x}_2 = 0.$

Поведение дискретной переменной l будет следующим:

$$l(v_0) = 0,\ l(v_1) = 1,\ l(v_2) = 2,\ l(v_3) = 3,$$

в предикатах эта переменная не участвует.

Переходы из одной локации в другую осуществляются при достижении критических значений температуры в реакторе:
условие перехода $v_0 \to v_1 :$ $\theta = \theta_M \wedge x_1 \geq T,$
условие перехода $v_0 \to v_2 :$ $\theta = \theta_M \wedge x_2 \geq T,$
условие переходов $v_1 \to v_0, v_2 \to v_0 :$ $\theta = \theta_m,$

Переход в локацию v_3 соответствует аварийной ситуации и осуществляется при невозможности использования обоих стержней:
условие перехода $v_1 \to v_3 :$ $\theta = \theta_M \wedge x_1 < T \wedge x_2 < T.$

Значения переменных на переходах гибридного автомата могут изменяться,в нашем случае переменные x_1 и x_2 должны обнуляться на переходах $v_1 \to v_0$ и $v_2 \to v_0$ соответственно, переменная l принимает номер текущей локации.

В качестве начальной локации выберем локацию 0 и определим вектор начальных значений переменных гибридного автомата как $\theta = \theta_m, x_1 = T, x_2 = T.$

• Описание для каждой локации гибридного автомата `tcp`-предикаты одной из предложенных формул (например, для асинхронной политики, см.формулу (***), сноска 5):

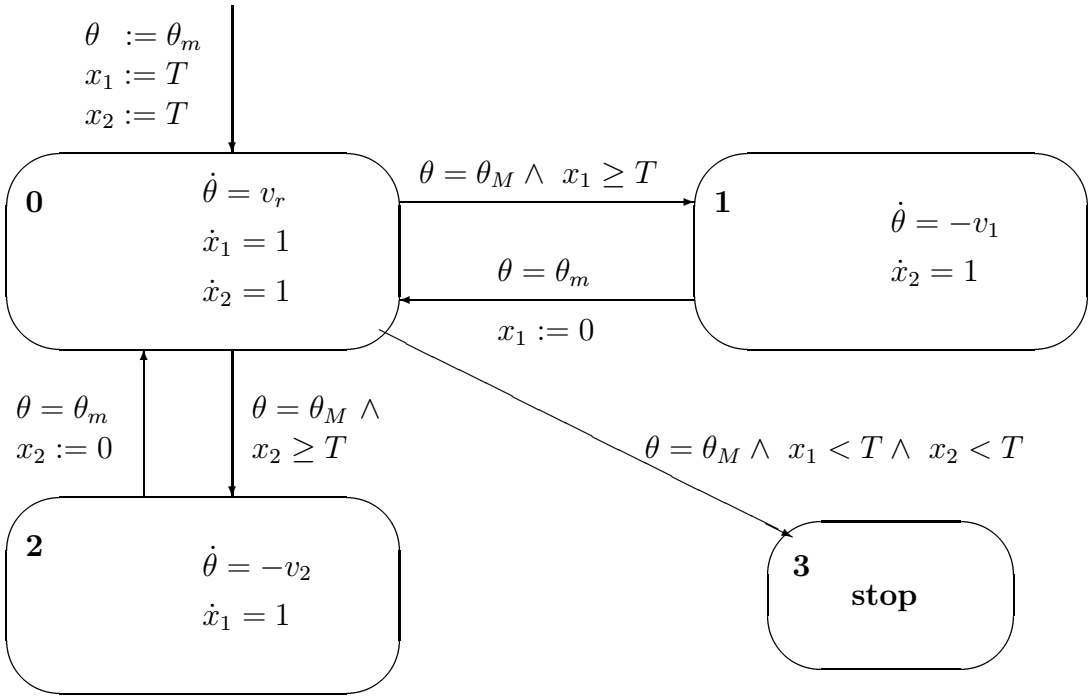


Рис. 4: Гибридный автомат системы температурного контроля атомного реактора.

$\texttt{tcp}_0[X_0](t) \equiv \theta_0 + v_r t \leq \theta_M \vee \theta_0 \geq \theta_M \quad \rightarrow \texttt{tcp}_0[X](t) \equiv \theta + v_r t \leq \theta_M$
$\texttt{tcp}_1[X_0](t) \equiv \theta_0 - v_1 t \geq \theta_m \vee \theta_0 \leq \theta_m \quad \rightarrow \texttt{tcp}_1[X](t) \equiv \theta - v_1 t \geq \theta_m$
$\texttt{tcp}_2[X_0](t) \equiv \theta_0 - v_2 t \geq \theta_m \vee \theta_0 \leq \theta_m \quad \rightarrow \texttt{tcp}_2[X](t) \equiv \theta - v_2 t \geq \theta_m$
$\texttt{tcp}_3[X_0](t) \equiv true \quad \rightarrow \texttt{tcp}_3[X](t) \equiv true.$

• Спецификация требования к поведению темпоральной формулой. Требование безостановочной работы реактора может быть описано требование корректности

$$Pr : I \Rightarrow \Box\neg(l = 3)$$

где I — предикат, описывающий начальные значения всех переменных и значения параметров гибридного автомата. В нашем случае

$$I \equiv \theta_M = 15 \wedge \theta_m = 3 \wedge v_r = 6 \wedge v_1 = 4 \wedge v_2 = 3 \wedge T = 6.$$

Для удобства преобразуем требование Pr в требование жизнеспособности:

$$\neg\exists\Diamond(l = 3)$$

• Применение одного из итерационных правил построения характеристического предиката для темпоральной формулы $P \equiv \exists\Diamond P_\varphi$ (где $P_\varphi \equiv l = 3$). Очевидно, следует выбрать правило 1 для формулы $P_{\varphi_1}\exists\mathcal{U}P_{\varphi_2}$, так как $\exists\Diamond P_\varphi \equiv true\exists\mathcal{U}P_\varphi$.

Характеристический предикат формулы P есть $\vee_i P_i$, где

$$P_0 = P_\varphi \equiv l = 3, \quad P_{i+1} = P_i \vee true \triangleright P_i$$

Заметим, что $(true \triangleright P)(X) \equiv$

$\exists e \exists t(\mathtt{pre}_e(P)(\varphi_v[X](t)) \wedge (\mathtt{tcp}_v[X](t)) \vee P(\varphi_v[X](t)) \wedge \mathtt{tcp}_v[X](t)$

кроме того, очевидно, что

$\mathtt{pre}_e(P_1 \vee P_2) = \mathtt{pre}_{e_1} P_1 \vee \mathtt{pre}_{e_2} P_2$,

где $\mathtt{pre}_{e_i} P = pre_{e_i}(X) \wedge P(post_{e_i}(X))$

Вычисление итерационных формул:

$P_1 \equiv (l = 3) \vee true \triangleright (l = 3) = (l = 3) \vee \exists e \exists t(l = 0) \wedge \theta + 6t = 15 \wedge x_1 + t < 6 \wedge x_2 + t < 6 \wedge \theta \leq 15) \vee (true \wedge l = 3) =$

$\{6t = 15 - \theta\} = (l = 0 \wedge \theta \leq 156x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta) \vee (l = 3);$

$P_2 = P_1 \vee true \triangleright P_1;$

$\mathtt{pre}_e[P_1 \vee P_0] = \mathtt{pre}_e[(l = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta) \vee (l = 3)] =$

$\mathtt{pre}_e(l = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta) \vee \mathtt{pre}_e(l = 3) = \mathtt{pre}_{e_{10}} P_1(X) \vee \mathtt{pre}_{e_{20}} P_1(X) \vee \mathtt{pre}_{e_{03}} P_0(X);$

$\mathtt{pre}_{e_{10}} P_1(X) \equiv \theta = 3 \wedge x_1 = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta;$

$\mathtt{pre}_{e_{20}} P_1(X) \equiv \theta = 3 \wedge x_2 = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta;$

$true \triangleright (P_1 \vee l = 3)(X) \equiv \{\theta^{(1)} = \theta - 4t; \theta^{(2)} = \theta - 3t; x_1 \to x_1 + t; x_2 \to x_2 + t\} =$

$l = 1 \wedge \theta - 4t = 3 \wedge x_1 = 0 \wedge \theta - 4t \leq 15 \wedge 6(x_2 + t) < 21 + (\theta - 4t) \wedge 6(x_1 + t) < 21 + (\theta - 4t) \wedge \theta \geq 3$

$\vee$

$l = 2 \wedge \theta - 3t = 3 \wedge x_2 = 0 \wedge \theta - 3t \leq 15 \wedge 6(x_2 + t) < 21 + (\theta - 3t) \wedge 6(x_1 + t) < 21 + (\theta - 3t) \wedge \theta \geq 3$

$\vee$

$l = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta$

$\vee$

$(l = 3) =$

$[l = 1 \wedge 3 \leq \theta \leq 15 \wedge 6x_2 + 10t < 21 + \theta \wedge \theta - 4t = 3] \vee$

$[l = 2 \wedge 3 \leq \theta \leq 15 \wedge 6x_1 + 9t < 21 + \theta \wedge \theta - 3t = 3] \vee$

$[l = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta] \vee [l = 3] =$

$\{4t = \theta - 3; \to 12x_2 + 3\theta < 57; 3t = \theta - 3 \to 6x_1 + 3\theta < 30\} =$

$[l = 1 \wedge 3 \leq \theta \leq 15 \wedge 4x_2 + \theta < 19] \vee$

$[l = 2 \wedge 3 \leq \theta \leq 15 \wedge 3x_1 + \theta < 15] \vee$

$[l = 0 \wedge \theta \leq 15 \wedge 6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta] \vee [l = 3]$;
аналогичным образом получаем,

$P_3 \equiv [l = 0 \wedge \theta \leq 15 \wedge (6x_1 < 21 + \theta \wedge 6x_2 < 21 + \theta \vee 6x_2 + 3 < \theta)] \vee$
$[l = 1 \wedge 3 \leq \theta \leq 15 \wedge 4x_2 + \theta < 19] \vee$
$[l = 2 \wedge 3 \leq \theta \leq 15 \wedge 3x_1 + \theta < 15] \vee [l = 3]$;
$P_4 \equiv P_3$

Предикат $\neg \bigvee_{i=0}^{3} P_i(0)$, представляющий множество предшествующих состояний, из которых достижимы состояния, описанные темпоральной формулой $\neg\exists\Diamond(l = 3)$, имеет вид:
$S \equiv\ l = 0 \wedge \theta \leq 15 \wedge (\theta + 21 \leq 6x_1 \wedge \theta \leq 6x_2 + 3 \vee \theta + 21 \leq 6x_2) \vee$
$l = 1 \wedge 3 \leq \theta \leq 15 \wedge 19 \leq 4x_2 + \theta \vee$
$l = 2 \wedge 3 \leq \theta \leq 15 \wedge 15 \leq 3x_1 + \theta$;

• Верификация требования к поведению. Для верификации требования к поведению Pr следует доказать, что $I \wedge [\neg\exists\Diamond(l = 3)] = I \wedge S \neq \emptyset$.

В данном случае видим, что $I \wedge S \neq \emptyset$, следовательно требование корректности доказано.

Нетрудно доказать, что при значении $T = 8$ требование корректности уже не будет удовлетворяться. Действительно, результатом итерационного алгоритма для темпоральной формулы $\neg\exists\Diamond(l = 3)$ после восьми итераций будет являться предикат $\neg \bigvee_{i=0}^{7} P_i(0)$:
$S \equiv l = 0 \wedge \theta > 15 \vee$
$l = 1 \wedge (\theta < 3 \vee \theta > 15) \vee$
$l = 2 \wedge (\theta < 3 \vee \theta > 15)$

Так как предикат $I \equiv l = 0 \wedge \theta = 3 \wedge x_1 = 8 \wedge x_2 = 8$, характеризующий множество начальных состояний системы, не удовлетворяет вышеописанному предикату S $(\theta > 15)$, то локация 3 гибридного автомата достижима [67]).

В работе [48] демонстрируется использование метода символьной верификации (с использованием системы автоматической верификации HyTech) для решения задачи параметрического анализа значений T, при которых система температурного контроля атомного реактора будет удовлетворять необходимому требованию. Задача решена в более общем случае, когда локальные поведения переменной θ описываются дифференциальными включениями.

1.6 Условия сходимости метода символьной верификации

Проблемой достижимости для гибридного автомата H называется доказательство существования непустого пересечения $Reach(H) \cap Z$, где $Reach(H)$ – характеристическое множество автомата, Z – некоторая заданная область его фазового пространства.

Проблемой пустоты ω-языка гибридного автомата H называется доказательство утверждения, что $Lang(H) \neq \emptyset$.

Проблема достижимости может быть решена путем верификации некоторого требования корректности. Проблема пустоты ω-языка является более общей, чем проблема достижимости, и сводится к верификации как требований корректности, так и требований жизнеспособности. Доказательство сходимости метода символьной верификации базируется на разрешимости проблем достижимости и пустоты ω-языка гибридного автомата, о именно: решение проблемы достижимости достаточно для доказательства сходимости итерационного алгоритма построения характеристического предиката темпоральных формул, описывающих класс свойств корректности в гибридном автомате, а решение проблемы пустоты ω-языка достаточно для доказательства сходимости итерационного алгоритма построения характеристического предиката темпоральных формул, описывающих все классы свойств поведения гибридного автомата [53].

Приведем несколько утверждений о сходимости метода символьной верификации:

Метод символьной верификации сходится для класса инициализированных прямоугольных автоматов с ограниченной интервальной инициализацией и всех классов свойств поведения. Метод символьной верификации сходится для класса инициализированных прямоугольных автоматов и свойств поведения корректности.

Доказательство этого утверждения приводится в работе [53] и основывается на преобразовании инициализированного прямоугольного автомата A в таймированный автомат B_A, которое обладает следующими свойствами:

$$Reach(A) = Reach(B_A) \quad \text{и} \quad Lang(A) \subset Lang(B_A)$$

Для таймированных автоматов известно, что их характеристические множества и ω-языки могут быть эффективно построены [68]. Следовательно, разрешимость проблемы достижимости (и сходимость метода символьной верификации для класса свойств корректности) для инициализированного прямоугольного автомата доказана. Однако, автомат B_A может допускать бесконечные расходящиеся слова, которые не допускает автомат A. Следовательно, проблема пустоты ω-языка автомата A (и сходимость метода символьной верификации для остальных классов свойств поведения) не может быть сведена к аналогичной проблеме в B_A. Однако, в той же работе [53] доказывается разрешимость проблемы пустоты ω-языка для инициализированного прямоугольного автомата с ограниченной интервальной инициализацией. Разрешимость проблемы пустоты ω-языка для инициализированного прямоугольного гибридного автомата, не обладающего свойством ограниченной интервальной инициализацией, на сегодняшний день не изучена.

Ряд работ гибридного направления посвящен доказательству того, что класс инициализированных прямоугольных автоматов является максимальным классом гибридных систем, для которого проблема достижимости разрешима.

Доказана неразрешимость проблемы достижимости для *простого* автомата (простым автоматом называется неинициализированный прямоугольный автомат размерности n, у которого только одна переменная не является таймером и имеет два возможных локальных поведения вида $\dot{x} = k_1,\ \dot{x} = k_2,\ k_i \in Q$, а все предикаты описывают компактные области в R^n) [53]. Это доказывает необходимость наличия свойства инициализации для разрешимости классов гибридных систем, которые отличаются от таймированного автомата одной (или более) переменной с двумя (или более) локальными поведениями.

Важную роль играет прямоугольный вид областей, описываемых предикатами автомата, и класс локальных поведений, ограниченный прямоугольными дифференциальными включениями. Усложнение этих компонент описания гибридного автомата приводит к неразрешимости проблемы достижимости.

Так, проблема достижимости не разрешима для простого автомата

размерности n с треугольными предикатами $pre, inv, init, post$, даже в случае, когда переменная x имеет единственное возможное локальное поведение $\dot{x} = k (k \neq 1)$ (треугольными предикатами называются линейные предикаты вида $x_i \leq x_j$, где x_i, x_j – переменные автомата, они описывают треугольные области в R^n) [53].

Проблема достижимости не разрешима для инициализированного гибридного автомата с тремя фазовыми переменными и с треугольными дифференциальными включениями (под треугольными дифференциальными включениями понимают локальные поведения, описываемые неравенствами вида $a \leq \dot{x}_i \leq \dot{x}_j \leq b,\ a, b \in Q,\ x_i, x_j$ – переменные автомата).

Однако, в вопросе исследования границ разрешимости вышеназванных проблем существует ряд противоречий.

Например, исключением является класс гибридных систем, у которых все фазовые переменные имеют кусочно–постоянные производные,а траектории являются ломаными линиями (нет разрывов) размерности 2. Несмотря на то, что этот класс не обладает свойством инициализированности, для него проблема достижимости полностью решена [34] (для этого класса гибридных систем размерности ≥ 3 проблема достижимости не разрешима).

Несмотря на то, что в работе [56] доказана неразрешимость проблемы достижимости для линейного гибридного автомата, для этого класса возможна успешная символьная верификация подмножества "неизбежных"(inevitable) формул логики деревьев вычислений (в этом случае гарантируется правильность положительных результатов верификации) [69].

Для линейных гибридных автоматов, представимых моделью интегрирующего графа, проблема достижимости оказывается также разрешимой[56].

Эти противоречия свидетельствуют о том, что требования инициализируемости и прямоугольности не являются всецело определяющими разрешимость проблем достижимости и пустоты ω-языка. В ряде последних работ, например в [52], вводятся новые характеристики, влияющие на разрешимость проблем достижимости и пустоты ω-языка, — *конечность* и *эффективность*. Гибридный автомат может

быть представлен системой переходов с бесконечным пространством состояний. Введем отношение эквивалентности между состояниями и будем производить разбиение пространства состояний на классы эквивалентности. Если отношение эквивалентности обладает свойством, что для каждого класса эквивалентности и для каждого перехода системы множество достижимых состояний является также одним классом эквивалентности, и множество начальных состояний относится к одному классу эквивалентности, то назовем это отношение "моделированием" (bisimulation) или моделирующим отношением эквивалентности для данной системы переходов. Если число классов эквивалентности конечно, то такое отношение эквивалентности называется конечным.

Система переходов называется *конечной*, если для нее существует конечное моделирующее отношение эквивалентности. Если разбиение на классы эквивалентности может быть эффективно построено (то есть может быть вычислено практически), то такая система переходов называется *эффективной*.

В теории бисимуляции гибридных систем доказывается, что проблемы достижимости и пустоты ω-языка полуразрешимы для эффективных систем переходов и полностью разрешимы для конечных и эффективных систем переходов. В частности, конечность и эффективность отношения эквивалентности доказана для класса линейных гибридных автоматов с постоянными целочисленными коэффициентами, который представляют собой более широкий класс, чем инициализированный прямоугольный гибридный автомат [69].

Таким образом, в теории бисимуляции доказано следующее утверждение [38]:

Метод символьной верификации сходится для класса класса линейных гибридных автоматов с постоянными целочисленными коэффициентами.

Таким образом, точные границы разрешимости гибридного подхода пока не получены. С введением в последнюю версию системы *HyTech* полуразрешимых процедур построения характеристических множеств гибридных автоматов с оценкой предела соответствующего итерационного процесса [73, 51] стало возможным использовать ее

для верификации линейных гибридных автоматов, но без гарантии их сходимости.

В настоящей работе мы будем использовать тот факт, что класс прямоугольных автоматов с ограниченной интервальной инициализацией и класс линейных гибридных автоматов с постоянными целочисленными коэффициентами может быть полностью верифицирован методом символьной верификации. Подробный обзор материалов по методам верификации содержится в работе автора [24].

1.7 Алгоритмы линейной аппроксимации гибридных автоматов

В данном разделе представлено направление разработки алгоритмов линейной аппроксимации, с помощью которых отдельные классы гибридных автоматов могут быть аппроксимированы моделью прямоугольного гибридного автомата (с целью их верификации символьным методом).

На сегодняшний день разработаны два алгоритма линейной аппроксимации – *таймер-преобразование (clock translation)* [54] и *δ–аппроксимация по производной (δ–approximate rate translation)* [49]. Они подробно представлены ниже.

1.7.1 Таймер–преобразование

Пусть дан гибридный автомат $A = \{X, V, E, \Sigma, flow, pre, post, init, inv\}$. Переменную гибридного автомата $x \in X$ будем называть *линейной*[6], если ее локальные поведения $flow(v, x)$ имеют вид $x_v(t) = x_v(0) + k_v t$, где $x_v(0) \in D$ (D – выпуклая область в R^n), $k_v = const$. Если хотя бы одно локальное поведение переменной не может быть представлено в таком виде, то такую переменную назовем *нелинейной*.

Идея таймер-преобразования заключается в аппроксимации нелинейных переменных таймерами.

[6]Понятия линейности в теории гибридных систем и в классической теории динамических систем не совпадают. Поведения линейных динамических систем (в классическом смысле) считаются сильно нелинейными локальными поведениями в гибридных автоматах

Пусть переменная $x \in X$ гибридного автомата отвечает следующим требованиям:

• локальное поведение $flow(v, x)$ переменной x в каждой локации v является решением дифференциального уравнения $\dot{x} = f_v(x)$;

• для каждой константы c, участвующей в предикате автомата с переменной x, функция "$x_v(t) - c$", где $x_v(t)$ – решение дифференциального уравнения $\dot{x} = f_v(x)$, имеет конечное число вещественных корней;

• для локаций v и v', если $flow(v, x) \neq flow(v', x)$ (локальные поведения переменной x в локациях различны) и существует дуга $e = (v, v')$, то на этой дуге имеется инициализация переменной вида "$x := c$"; множество начальных значений этой переменной $init(x)$ задается множеством точек;

• предикаты автомата $pre(e, x), inv(v, x)$ имеют вид "$x \sim c$", где c – конечное вещественное число, " $\sim$ " $= \{<, \leq, =, \geq, >\}$.

Конечное упорядоченное множество констант $c_1 < \ldots < c_n$, которые участвуют в инициализациях и предикатах на дугах автомата с этой переменной, назовем множеством $CritVal(x)$ ее критических значений.

Опишем алгоритм преобразования переменной x в таймер (таймер-преобразование).

Алгоритм таймер-преобразования:

Шаг 1. Рассмотрим локацию v гибридного автомата. Разобьем локацию на конечное множество подлокаций $\{v_{c_1}, \ldots v_{c_n}\}$, каждая из которых соответствует одному из критических значений переменной. Значение переменной x в подлокации v_{c_i} будет являться решением $x_v(t)$ задачи Коши с соответствующими начальными условиями "$\dot{x} = f_v(x); x(0) = c_i$".

Шаг 2. Заменим переменную x на таймер t_x (локальные поведения $flow(v_{c_i}, t_x) : \dot{t}_x = 1$ во всех подлокациях). Следует описать правила наследования дуг автомата, условия переходов, инициализацию таймера на новых дугах и инварианты в подлокациях.

Преобразование множеств $E, init$ и $post$. Дуге $e = (v, v')$, на которой не было инициализации переменной x, соответствует множество дуг (v_{c_i}, v'_{c_i}), каждая из которых наследует метку дуги $label(e)$,

условие перехода по другим переменным и имеет инициализацию $post(e) \wedge t'_x = t_x$ (штрихом помечается значение переменной на конце дуги). Дуге $e = (v, v')$ с инициализацией переменной $x := c_j$ соответствует множество дуг из всех подлокаций v в подлокацию v_{c_j}, каждая из которых наследует метку дуги $label(e)$, условие перехода по другим переменным и имеет инициализацию $post(e) \wedge t'_x = 0$. Очевидно, часть локаций после этой процедуры следует удалить (те, у которых отсутствуют входные дуги). Следует удалить также подлокации, соответствующие критическим значениям, не удовлетворяющим инварианту исходной локации, например, если $c_i > c$ и инвариант исходной локации $x \leq c$, то подлокация v_{c_i} удаляется.

Преобразование множества pre. Пусть теперь $x_{v_{c_i}}(t)$ решение задачи Коши "$\dot{x} = f_v(x); x(0) = c_i$". После замены переменной на таймер все термы в предикатах автомата, включающие x, удаляются. Предположим, что $x \leq c$ — терм на некоторой дуге, исходящей из подлокации v_{c_i}. Это условие перехода необходимо заменить на ограничение, накладываемое на значение таймера t_x. Для этого находятся все неотрицательные вещественные корни ($r_0, \ldots, r_k$, в соответствии с требованиями их конечное число) уравнения

$$x_{v_{c_i}}(t) - c = 0 \tag{9}$$

Если корней не существует и $c_i \leq c$, то условие $x \leq c$ удовлетворяется всегда и условие $x \leq c$ заменяется на $True$; если корней не существует и $c_i > c$, то условие $x \leq c$ заменяется на $False$. При условии существования корней если $c_i \leq c$, то терм $x \leq c$ заменяется ограничением на значения таймера t_x в виде дизъюнкции интервалов $[0; r_0], [r_1, r_2], \ldots$; если $c_i > c$, то терм $x \leq c$ заменяется ограничением на значения таймера t_x в виде дизъюнкции интервалов $[r_0; r_1], [r_2; r_3], \ldots$.

Преобразование множества inv. Если "$x \leq c$" – терм инварианта подлокации v_{c_i} и $c_i \leq c$, то находим наименьший неотрицательный корень r уравнения $x_{v_{c_i}}(t) - c = 0$ и если такой корень существует, то заменяем "$x \leq c$" на терм $t_x \leq r$, а если нет, то на $True$. □

Алгоритм таймер-преобразования был предложен П.-Т.Хо [54] и

реализован в системе HyTech. Вычислительная сложность таймер-преобразования равна вычислительной сложности решения $n \cdot m(x)$ задач нахождения всех нулей функций одной вещественной переменной (n – число нелинейных переменных автомата, $m(x)$ – число критических значений переменной x). Число локаций в преобразованном автомате не превышает $n \cdot m(x) \cdot p$ (p – количество локаций исходного автомата).

Пусть теперь все нелинейные переменные удовлетворяют вышеперечисленным требованиям. Применим к каждой из них таймер-преобразование и обозначим построенный автомат через B_A. В работе [54] доказано следующее утверждение о таймер-преобразовании:

Теорема 1.7.1 *Поведение автомата B_A эквивалентно поведению исходного автомата A:*

$$Reach(A) = Reach(B_A), \quad Lang(A) = Lang(B_A).$$

Заметим, что построенный автомат B_A является (при наличии других линейных переменных) линейным гибридным автоматом и может быть верифицирован в системе HyTech.

Если линейных переменных нет, а все нелинейные переменные удовлетворяют требованиям таймер-преобразования, такой гибридный автомат преобразуется в таймированный автомат, для которого применимы также методы автоматической верификации систем реального времени [68].

1.7.2 Алгоритм δ–аппроксимации по производной

Пусть дан гибридный автомат $A = \{X, V, E, \Sigma, flow, pre, post, init, inv\}$ и пусть $\delta \in R_{\geq 0}$ — неотрицательное вещественное число.

Переменную x гибридного автомата A назовем *ограниченной интервалом* $[c; d] \in R$, если выполнено одно из трех следующих требований:

- значение x всегда находится в ограниченном интервале $[c; d]$. Например, это требование выполнено в случае, если инварианты во всех локациях автомата ограничивают x в пределах $[c; d]$

• автомат A не содержит констант меньше c (больше d) и переменная x неубывающая (невозрастающая) во всех локациях A

• верифицируемые требования к поведению автомата содержат требование корректности вида $\Box(c \leq x \leq d)$.

Идея алгоритма заключается в замене нелинейной переменной ее кусочно-линейной аппроксимацией.

Алгоритм δ–аппроксимации автомата A:

Шаг 1. (одномерный случай) Рассмотрим одну нелинейную ограниченную переменную x локальными поведениями $flow(v, x) : \dot{x} = f_v(x)$. Интервал $[c; d]$ разобьем на k подынтервалов длиной не больше δ (наиболее выгодным является разбиение, при котором границы подынтервалов совпадают с критическими значениями переменной x). Каждую локацию v разобьем соответственно на $k + 2$ подлокации $v_0, \ldots, v_{k+1}$, каждая из которых имеет свой инвариант вида $inv(v) \wedge\ c_{i-1} \leq x \leq c_i,\ c_{-1} = -\infty, c_{k+1} = +\infty$. Для каждой подлокации вычисляем минимум a и максимум b функции $f_v(v_i, x)$ в интервале $c_{i-1} \leq x \leq c_i$ и локальное поведение переменной в этой подлокации заменяем на дифференциальное включение $flow(v_i, x) : \dot{x} \in [a; b]$.

Дуги автомата модифицируются следующим образом. Каждая дуга автомата $e = (v, v')$ заменяется множеством дуг вида (v_i, v'_j) с наследованием метки, условия перехода и инициализации. Кроме того, подлокации одной локации соединяются парами дуг $(v_i, v_{i+1}), (v_{i+1}, v_i)$ с условием переходов $x = c_i$.

Шаг 2.(n-мерный случай) Рассмотрим n нелинейных переменных, ограниченных интервалами $I_1, \ldots, I_n$. Пусть функция локального поведения нелинейной переменной x_i в локации v, имеет вид $flow(v, x_i) : \dot{x}_i = f_v(x_1, \ldots, x_n)$. Будем последовательно аппроксимировать все нелинейные переменные. Интервал I_i разбиваем на k подынтервалов, каждый из которых длиной не больше δ. Локация v разбивается на множество подлокаций $U_v = \{v(a_1, \ldots, a_n)\}$,где $a_i \in \{0, 1, \ldots, k_i{+}1\}$ обозначает номер подынтервала для каждой переменной (всего $(k_1 + 2) \cdots (k_n + 2)$ подлокаций). Инвариант подлокации $v(a_1, \ldots, a_n)$ есть $inv(v(a_1, \ldots, a_n)) : inv(v)\ \wedge\ \wedge_{i=1,\ldots,n} c_{i,a_i-1} \leq x \leq c_{i,a_i}$ где $c_{i,-1} = -\infty, c_{i,k_i+1} = +\infty$. В каждой подлокации вычисляются дифференциальные включения для всех переменных $x_1, \ldots, x_n$ по схеме,

описанной в шаге 1. Дуги наследуются по аналогии с одномерным случаем. □

Алгоритм δ–аппроксимации по производной был предложен Т.Хензингером и П.-Т.Хо [49].

Будем называть построенный автомат δ–аппроксимацией по производной автомата A (и обозначать $[A^r]_\delta$). Количество локаций в нем равно $p \cdot (k_1 + 2) \cdots (k_n + 2)$ (p – число локаций исходного автомата). Вычислительная сложность δ–аппроксимации соответствует вычислительной сложности решения $p \cdot k(\delta)$ задач нахождения экстремальных значений функций одной вещественной переменной (одномерный случай) или $p \cdot k_1(\delta) \cdots k_n(\delta)$ задач нахождения экстремальных значений функции n вещественных переменных (n–мерный случай, p – количество локаций исходного автомата).

В работе [54] доказано следующее утверждение об алгоритме δ–аппроксимации по производной:

Теорема 1.7.2 *Линейное поведение автомата $[A^r]_\delta$ подобно линейному поведению исходного автомата A:*

$$Reach(A) \subseteq Reach([A^r]_\delta), \quad Lang(A) \subseteq Lang([A^r]_\delta).$$

Любая линейная последовательность, допущенная автоматом A, допускается и автоматом $[A^r]_\delta$, но не наоборот.

В случае, если предикаты исходного автомата описывают прямоугольные области, полученный автомат будет являться прямоугольным автоматом, для которого проблема достижимости разрешима.

Таким образом, данная теорема является обоснованием применимости метода символьной верификации для верификации тех свойств поведения, результат верификации которых есть $True$, для гибридных автоматов с нелинейными ограниченными переменными.

1.7.3 Пример: термостат

Рассмотрим пример линейной аппроксимации гибридного автомата с одной нелинейной переменной.

Пусть температура в комнате должна поддерживаться в определенных пределах с помощью термостата, который непрерывно изменяет температуру и может находиться в двух основных режимах работы — в режиме нагрева "heater on"и в режиме ожидания "heater off". Каждому режиму работы соответствует свое дифференциальное уравнение, которое описывает изменение температуры. Обозначим температуру переменной x. Режиму нагрева соответствует дифференциальное уравнение $\dot{x} = K(h - x)$, где K – константа, характеризующая параметры комнаты, h – константа, характеризующая мощность термостата. Температура в комнате изменяется по закону $x(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$, где t – время, θ – начальное значение температуры при включении нагрева. Режиму ожидания соответствует дифференциальное уравнение $\dot{x} = -Kx$ и температура в комнате изменяется по закону $x(t) = \theta e^{-Kt}$, где θ – начальное значение температуры при выключении нагрева. Обозначим граничные значения температуры через m и M соответственно.

Опишем поведение системы гибридным автоматом с единственной нелинейной переменной x. Каждому режиму работы системы поставим в соответствие локацию гибридного автомата (локация 1 – режим "heater on", локация 2 – режим "heater off"). Переходы из одной локации в другую осуществляются при достижении температурой критических значений:

условие перехода $v_1 \to v_2$: $\quad x = M$,

условие перехода $v_2 \to v_1$: $\quad x = m$.

Для определенности будем считать, что начальная температура меньше M, поэтому выберем локацию 1 в качестве начальной локации. Гибридный автомат для системы управления термостатом изображен на рисунке 5.

Предположим теперь, что мы хотим верифицировать требование к поведению, отражающее, что термостат находится в режиме ожидания не менее половины общего времени, и получить оценку отношения времени нагрева к общему времени работы термостата.

Для верификации требования и решения задачи параметрического анализа введем в гибридный автомат дополнительную вещественную переменную y (отслеживающую суммарное время нахождения во ре-

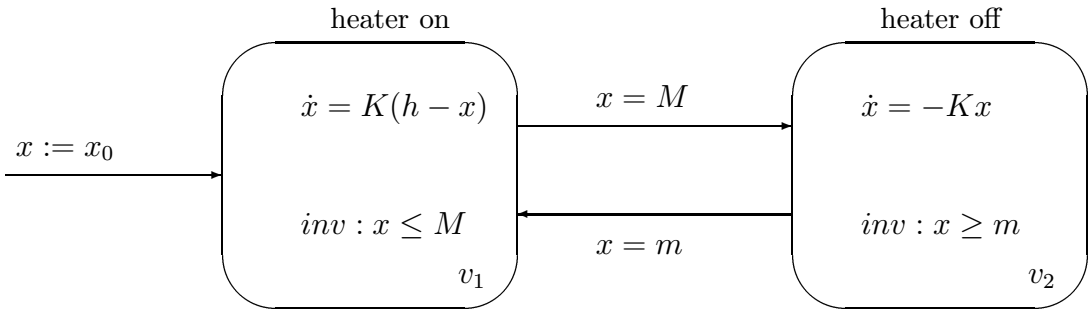


Рис. 5: Гибридный автомат системы управления термостатом.

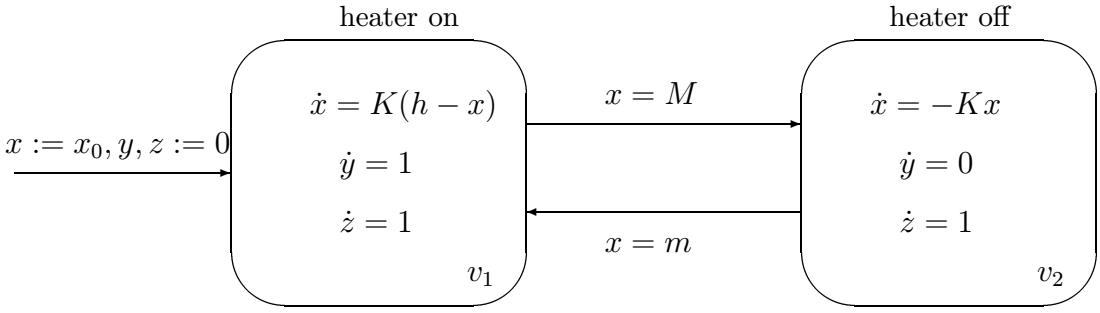


Рис. 6: Гибридный автомат для верификации системы управления термостатом.

жиме нагрева) и таймер z (отслеживающий общее системное время). Их локальные поведения опишутся следующим очевидным образом:

в локации $v_1 : \dot{y} = 1, \dot{z} = 1$;

в локации $v_2 : \dot{y} = 0, \dot{z} = 1$.

В предикатах эти переменные не участвуют и на переходах не инициализируются. Гибридный автомат теперь будет иметь вид, изображенный на рис. 6.

Верифицируемое требование может быть сформулировано как требование корректности и описано логической формулой: $z \geq t_0 \wedge 2y \leq z$, где t_0 определяет момент времени, начиная с которого требование можно проверять.

Как мы видим, гибридный автомат содержит нелинейную переменную, поэтому он не относится к классам автоматов, для которых

сходимость символьного метода верификации гарантирована. Однако, нелинейная переменная удовлетворяет всем требованиям таймер-преобразования.

Зададим конкретные значения параметров системы $m = 1, M = 3, K = 1, h = 5, x_0 = 2$ и произведем замену нелинейной переменной x на таймер t_x.

Множество критических значений для переменной x есть $CritVal(x) = \{1, 2, 3\}$.

На первом шаге алгоритма необходимо разбить каждую локацию на 3 подлокации (по количеству критических значений переменной x). Подлокации (heater on, 3),(heater off, 1) и (heater off, 2) оказываются недостижимыми (то есть не имеют входных дуг) из начальной локации (heater on, 2), поэтому могут быть удалены.

В результате шага 1 получим автомат, изображенный на рис.7,а.

На втором шаге алгоритма необходимо добавить в автомат таймер t_x и удалить переменную x, модифицировав все предикаты исходного автомата $pre, init, inv$.

Для каждой подлокации решение задачи Коши имеет вид:

в локации v_1 (heater on,2) : $x(t) = -3e^{-t} + 5$,

в локации v_2 (heater on,1) : $x(t) = -4e^{-t} + 5$,

в локации v_3 (heater off,3) : $x(t) = -3e^{-t}$.

Рассмотрим предикат $x = 3$ и локацию v_1. Так как уравнение $-3e^{-t} + 5 = 3$ имеет единственный корень $t = \ln(3/2)$, то предикат $x = 3$ следует заменить на предикат $t_x = \ln(3/2)$. Аналогичным образом заменяются предикаты $x = 3$ в локации v_2 и предикат $x = 1$ в локации v_3. На всех переходах автомата таймер t_x должен быть инициализирован $t_x = 0$.

В результате шага 2 таймер-преобразования получим автомат, изображенный на рис.7,б.

Этот автомат может быть легко преобразован в линейный гибридный автомат с постоянными коэффициентами и далее успешно верифицирован методом символьной верификации. Для этого достаточно заменить предикаты автомата вида "$t_x = \ln$ r" на рациональные неравенства вида $q_1 \le t_x \le q_2$, например, предикат $t_x = \ln 2$ должен быть замене на $69/100 \le t_x \le 70/100$ ($\ln 2 \approx 0.693$).

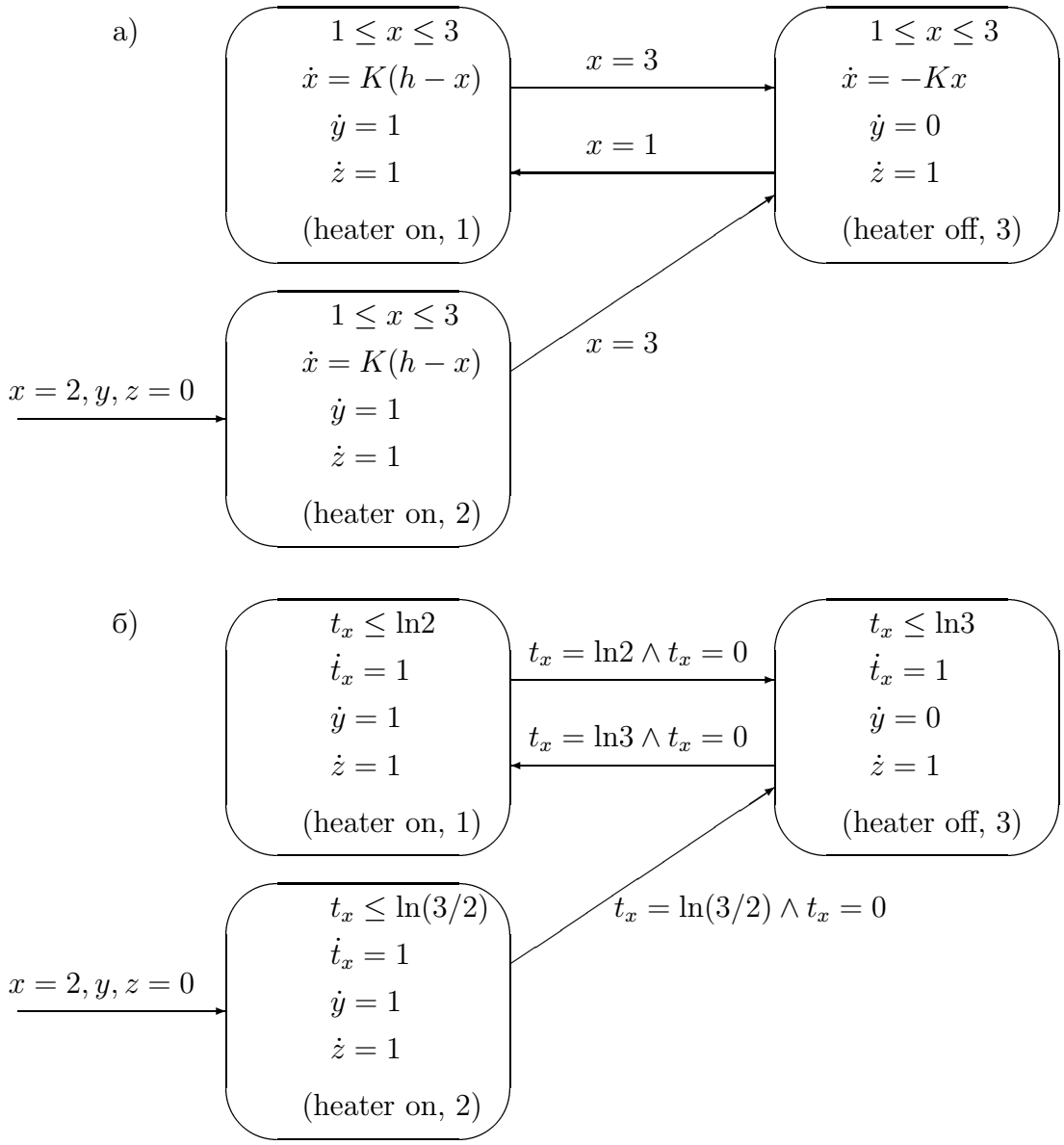


Рис. 7: Таймер-преобразование системы управления термостатом.

В работе [49] приводится результат решения задачи параметрического анализа для термостата в системе автоматической верификации HyTech — в режиме нагрева термостат находится от $(2317/60) \approx 38.6\%$ до $(2351/60) \approx 39.2\%$ от времени работы.

Линеаризируем теперь гибридный автомат системы управления термостатом (рис.5) с помощью алгоритма δ-аппроксимации по производной (имеем одномерный случай). Это возможно, так как переменная x ограничена интервалом $[1, 3]$. Возьмем δ=1. Каждая локация разбивается на 2 подлокации:

"heater on" $\rightarrow \{$("heater on$_1$", $1 \le x \le 2$)("heater on$_2$", $2 \le x \le 3$)$\}$,

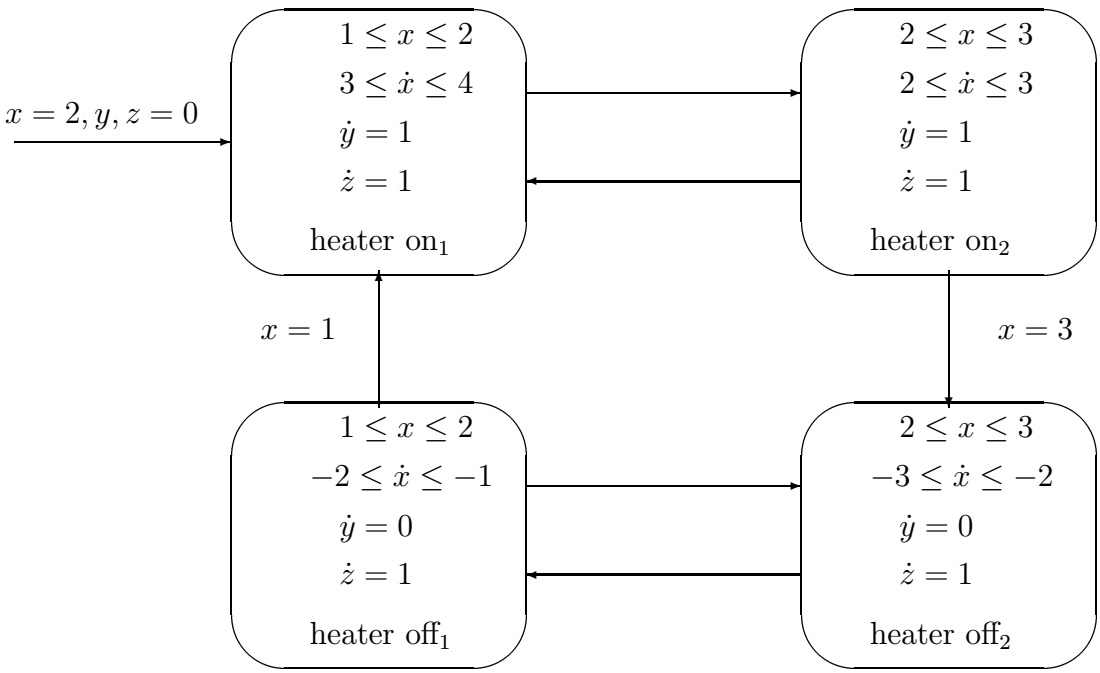


Рис. 8: δ-аппроксимация системы управления термостатом.

"heater off" $\rightarrow \{$("heater off$_1$", $1 \le x \le 2$)("heater off$_2$", $2 \le x \le 3$)$\}$.

Заменим в каждой подлокации дифференциальное уравнение на дифференциальное включение по стандартной схеме кусочно-линейной аппроксимации. Результирующий автомат изображен на рис.8.

В той же работе [49] приводится результат решения задачи параметрического анализа для термостата, с использованием алгоритма аппроксимации по производной, в системе автоматической верификации HyTech, — в режиме нагрева термостат находится от $\approx 30.7\%$ до $\approx 48.1\%$ от времени работы.

Этот результат демонстрирует, насколько более точная оценка получается при использовании таймер-преобразования по сравнению с аппроксимацией по производной при одинаковом порядке общего числа подлокаций.

1.7.4 Недостатки алгоритмов линейной аппроксимации

Следует отметить, что, несмотря на свою простоту и элегантность, оба вышеназванных алгоритма имеют недостатки.

Таймер-преобразование позволяет аппроксимировать очень узкий класс гибридных автоматов, поскольку обращение функций локального поведения производится аналитически. Напомним, что оно применимо только к отдельным нелинейным переменным гибридного автомата, если: 1) ее локальное поведение, задаваемое дифференциальным уравнением, имеет алгебраическое решение $x(t)$, такое что для всех констант c, участвующих в предикатах автомата с этой переменной, функция "$x(t) - c$" имеет конечное число вещественных корней; 2) при изменении функции локального поведения, а также в начальных вершинах автомата имеется детерминированная (точечная) инициализация переменной.

Кусочно-линейная аппроксимация (δ–аппроксимация по производной) применима для широкого класса гибридных автоматов с локальными поведениями, заданными системами дифференциальных уравнений $flow(v, x_i) : \dot{x}_i = f(x_1, \ldots, x_n), i = 1, \ldots, n$, у которых все переменные обладают свойством ограниченности. Однако вычислительная сложность δ–аппроксимации и общее количество подлокаций в построенном автомате полиномиально зависит от величины δ. Такая зависимость в практическом отношении сильно снижает привлекательность алгоритма. Так, авторы работ [72], [50] с прискорбием замечают, что разрастание локаций приводит к невозможности моделирования поведения большинства практических задач в системе HyTech. Кроме того, алгоритм δ–аппроксимации по производной позволяет верифицировать те требования к поведению исходной гибридной системы, результат верификации которых положителен ($True$). Путем уменьшения величины δ иногда удается верифицировать также требования к поведению, результат которых отрицателен (провести так называемый анализ "ошибочных траекторий", error analysis). В общем же случае полная верификация произведена быть не может.

Кусочно–линейная аппроксимация, примененная формально[7], име-

[7]В некоторых работах, например в [49], [72] перед применением δ-аппроксимации проводятся дополнительные неформализованные исследования фазового портрета системы, на основе которых "из общих соображений" сужают области возможных значений параметров системы, что позволяет избежать разрастания локаций. Очевидно, однако, что если верификация системы требует предварительного исследования ее фазового портрета, то она не имеет особого смысла.

ет, таким образом, следующие недостатки:

• часто дает слишком размытые границы дифференциальных включений в получаемом прямоугольном автомате;

• приводит к разрастанию локаций, при котором автомат не может быть эффективно верифицирован имеющимися современными техническими средствами;

• позволяет верифицировать лишь требования к поведению, результат верификации которых положителен.

Из вышесказанного следует, что существующие алгоритмы линейной аппроксимации не позволяют в достаточной мере использовать символьный подход в реальных задачах моделирования и анализа непрерывно–дискретных систем. Один из возможных путей разрешения этой проблемы заключается, по нашему мнению, в значительном расширении набора алгоритмов линейной аппроксимации гибридных автоматов.

2 Алгоритм обобщенного таймер-преобразования для гибридных автоматов с линейными системами ОДУ

В данной главе представлен новый алгоритм линейной аппроксимации гибридных автоматов — обобщенное таймер-преобразование.

Предложен метод покоординатных оценок для решения задачи оценивания фазового состояния линейных систем ОДУ с постоянными коэффициентами и граничными условиями, рассмотрены частные случаи решения.

Сформулированы и теоретически обоснованы условия применимости алгоритма обобщенного таймер-преобразования для гибридного автомата с линейными системами дифференциальных уравнений с постоянными коэффициентами и ограниченной интервальной инициализацией. Доказана теорема о сходимости метода символьной верификации для линейной аппроксимации гибридного автомата с линейными системами дифференциальных уравнений, построенной с помощью обобщенного таймер-преобразования, и для линейной аппроксимации параллельной композиции гибридных автоматов с линейными системами дифференциальных уравнений. Доказана теорема о корректности положительных результатов символьной верификации для гибридных автоматов с линейными системами дифференциальных уравнений и их параллельной композиции.

Таким образом, расширен класс непрерывно-дискретных систем, для которых возможна символьная верификация.

2.1 Гибридный автомат с линейными системами ОДУ

Рассмотрим линейную систему дифференциальных уравнений

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} a_{ij}(t)x_j + f_i(t), \quad x_i(0) = c_i, \ i = 1, \ldots, n. \tag{10}$$

где x_i – переменные системы, $a_{ij}(t), f_i(t)$ – вещественные функции, c_i – вещественные константы. Будем записывать (10) в матричной форме

$$\frac{d\mathbf{x}}{dt} = A(t)\mathbf{x} + \mathbf{f}(t), \quad \mathbf{x}(0) = \mathbf{c}$$

где $A = [a_{ij}(t)], i, j = 1 \ldots n$ – матрица размерности $n \times n$, $\mathbf{f} = [f_i(t)], i = 1 \ldots n$ – вектор-функция размерности n, $\mathbf{x} = [x_i]$ – вектор переменных системы.

Введем новый класс гибридных автоматов, у которых локальные поведения фазовых переменных описываются линейными системами дифференциальных уравнений, а все предикаты являются линейными. Будем называть их *гибридными автоматами с линейными*

системами дифференциальных уравнений (с линейными системами ОДУ).

Определение 2.1.1 *Гибридным автоматом с линейными системами дифференциальных уравнений называется гибридный автомат:*

$$H_{ОДУ} = \{\ \Sigma,\ V,\ X,\ E,\ flow,\ pre,\ post,\ inv,\ init,\ syn\ \},$$

где

Σ *— конечный входной алфавит (алфавит событий);*

V *— конечное множество вершин автомата (локаций);*

X *— конечное множество фазовых переменных (будем представлять его в виде вектора* $\mathbf{x} = [x_1, \ldots, x_n], x_i \in X$*);*

E *— конечное множество дуг;*

$syn : E \to \Sigma$ *– заданное конечное отображение;*

$flow$ *— конечное множество локальных поведений фазовых переменных вида:*

$$flow(v, \mathbf{x}) : \frac{d\mathbf{x}}{dt} = A_v\mathbf{x} + \mathbf{b}_v$$

где A_v *– матрица линейной системы ОДУ с постоянными коэффициентами в локации* v, $\mathbf{b}_v$ *– вектор постоянных коэффициентов неоднородной системы ОДУ в локации* v, $a_{ij}, b_i \in R$*;*

pre *– конечный набор логических формул над множеством фазовых переменных вида:*

$$pre(e)\ :\ \bigwedge_i\ p_i$$

где $p_i = \sum_{j=1}^{n} h_j x_j \sim c$ *– термы,* $h_j, c \in R, \sim = \{<, \leq, =, \geq, >\}, e \in E$*;*

$post$ *– набор мгновенных действий (инициализаций) при смене локальных поведений вида:*

$$post(e)\ :\ \{x_i := c_i,\ i \in 1 \ldots n\},\quad c_i \in [l_i; u_i],\quad l_i, u_i \in R,\ e \in E;$$

inv *— конечное множество логических формул (предикатов-инвариантов), описывающих дополнительные ограничения на значения фазовых переменных в локациях вида:*

$$inv(v)\ :\ \bigwedge_i\ p_i$$

где $p_i = \sum_{j=1}^{n} h_j x_j \sim c$ *– термы,* $h_j, c \in R, \sim = \{<, \leq, =, \geq, >\}, v \in V$*;*

init – предикат, описывающий множество начальных локаций и области начальных значений фазовых переменных:

$$init(v) \ : \ \{x_i := c_i, \ i = 1 \ldots n\} \ \vee \ False, \ \ c_i \in [l_i; u_i], \ l_i, u_i \in R, \ v \in V;$$

Каждая дуга $e \in E$ помечена буквой алфавита событий $a = syn(e)$, условием перехода по дуге $pre(e)$ и формулой преобразования фазовых переменных на дуге $post(e)$.

Будем обозначать через $\mathcal{R}$ множество всевозможных ограниченных вещественных интервалов, а через $\mathcal{P}$ – множество всевозможных ограниченных прямоугольных областей (прямоугольных параллелепипедов) в R^n.

Назовем дугу e, входящую в локацию v гибридного автомата, "синхронизированным входом" локации v, если формула инициализации переменных на ней $post(e)$ имеет вид:

$$post(e) : \{x_i := c_i, \ i = 1 \ldots n\}, \ c_i \in \mathcal{I}_i, \ \ \mathcal{I}_i \in \mathcal{R}.$$

Все фазовые переменные автомата при входе в локацию v по синхронизированному входу имеют заданные значения из ограниченного интервала; множество входных значений $\{\mathbf{x}_0^{(v)}\}$ локации v является прямоугольным параллелепипедом в R^n, $\{\mathbf{x}_0^{(v)}\} = D \in \mathcal{P}$.

Назовем дугу e, выходящую из локации v гибридного автомата, "синхронизированным выходом" локации v, если условие перехода по ней $pre(e)$ имеет вид:

$$pre(e) : \bigwedge_{i=1}^{n} a_i \le x_i \le b_i, \ \ [a_i, b_i] \in \mathcal{R}.$$

Все фазовые переменные автомата на выходе из локации v по синхронизированному выходу имеют заданные значения из ограниченного интервала, множество выходных значений $\{\mathbf{x}^{(v)}\}$ локации v является прямоугольным параллелепипедом в R^n, $\{\mathbf{x}^{(v)}\} = D \in \mathcal{P}$.

Будем обозначать дугу, которая является синхронизированным входом или синхронизированным выходом, через $Synch[D]$, где $D \in \mathcal{P}$ – заданная прямоугольная область.

Произвольный цикл гибридного автомата будем называть "синхронизированным", если хотя бы одна локация, участвующая в нем, имеет синхронизированный вход или синхронизированный выход.

Для несинхронизированных циклов гибридного автомата определим свойство "неразрастания" фазовых траекторий. Будем говорить, что при обходе несинхронизированного цикла пучок фазовых траекторий не разрастается, если для каждой локации v, участвующей в цикле, множество входных значений $\{\mathbf{x}_0^{(v)}\}$ обладает свойством:

$$\{\mathbf{x}_0^{(v)}\}_{\text{после обхода}} \subseteq \{\mathbf{x}_0^{(v)}\}_{\text{до обхода}}.$$

Будем говорить, что гибридный автомат с линейными системами ОДУ обладает свойством ограниченной интервальной инициализации, если

1) предикат начальных значений $init$ описывает все области начальных значений в виде ограниченных прямоугольных областей;

2) для каждой дуги автомата e соответствующее множество $post(e)$ имеет вид: $post(e) : \{x_i := c_i,\ i \in 1 \dots n\},\ c_i \in \mathcal{I}_i,\ \ \mathcal{I}_i \in \mathcal{R}$.

2.2 Обобщенное таймер–преобразование

Пусть дан гибридный автомат с линейными системами дифференциальных уравнений $H = \{\Sigma, V, X, E, flow, pre, post, inv, init, syn\}$.
Пусть автомат H обладает свойством ограниченной интервальной инициализации. Предположим также, что все его несинхронизированные циклы обладают свойством неразрастания фазовых траекторий.

Поставим задачу его символьной верификации.

Класс гибридных автоматов с линейными системами дифференциальных уравнений не относится к классам гибридных автоматов, для которых метод символьной верификации сходится (см.п.1.6). Поэтому будем решать задачу символьной верификации следующим образом:

1) построим линейную аппроксимацию гибридного автомата;

2) модифицируем требования к поведению для построенной модели поведения;

3) верифицируем построенную модель методом символьной верификации.

Для построения линейной аппроксимации предлагается алгоритм обобщенного таймер-преобразования.

2.2.1 Схема алгоритма.

На вход алгоритма обобщенного таймер-преобразования подается гибридный автомат H с линейными системами ОДУ. С помощью алгоритма строится автомат T_H, который является линейной аппроксимацией исходного автомата.

Основные шаги алгоритма заключаются в следующем:

Шаг 1. Для произвольной локации v гибридного автомата H вычисляем аппроксимации всех множеств возможных входных значений вектора фазовых переменных. Для этого будем идти по всем путям, ведущим в локацию v, до первого синхронизированного входа или синхронизированного выхода (либо мы дойдем до начальной локации автомата, либо до локации, участвующей в цикле, проходящем через v, с известными входными или выходными значениями всех компонент вектора фазовых переменных, либо вернемся в исходную локацию по несинхронизированному циклу, в последнем случае помечаем цикл) и отслеживать развитие $\mathbf{x}$ (пучка траекторий) вплоть до локации v путем последовательного решения задач оценивания фазового состояния линейных систем ОДУ с прямоугольными областями начальных значений.

Шаг2. Разбиваем локацию v на подлокации $\{v_1, \ldots, v_k\}$ (k – число различных множеств входных значений $\mathbf{x}$ в локации v, в худшем случае $k = c_0 + c_v$, где c_0 – количество начальных локаций автомата, c_v – количество циклов, проходящих через v), каждая из которых имеет свою область входных значений $\mathbf{x}_0^{(v_i)} = D^{(v_i)} \in \mathcal{P}$. Каждая новая подлокация v_i соединяется дугами со всеми такими локациями v', что в исходном автомате существовали дуги $e(v, v')$, и с одной локацией v'', такой что в исходном автомате существовала дуга $e(v'', v)$ и именно по ней шло вычисление входного значения $\mathbf{x}_0^{(v_i)}$ подлокации v_i (с наследованием предикатов $pre(e)$ и $post(e)$). Следует объединить подлокации с одинаковыми областями входных значений и удалить подлокации, в которых начальные значения не удовлетворяют инварианту $\mathbf{x}_0^{(v_i)} \notin inv(X, v)$.

Повторяем шаг 1 и шаг 2 последовательно для всех локаций автомата.

Шаг 3. Вектор фазовых переменных заменяем на (единственный) таймер t_X, который запускается во всех подлокациях (предикаты

$post(e)$ заменяются на $t_X := 0$). Предикаты $pre(e)$ на выходящих дугах из локации v и предикаты-инварианты $inv(v)$ заменяются на ограничения по длительности нахождения в локации v_i вида $t_1^{(v_i)} \le t \le t_2^{(v_i)}$, где значения $t_1^{(v_i)}, t_2^{(v_i)}$ берутся из решений задач оценивания фазового состояния для подлокации v_i и условия перехода по дуге $pre(e)$. □

Условие неразрастания несинхронизированных циклов автомата можно проверять по ходу работы алгоритма, исследуя каждое из всех вычисленных множеств возможных входных значений локации v на неразрастание при обходе помеченных циклов путем последовательного решения задач оценивания для локаций, участвующих в циклах.

Вычислительная сложность алгоритма равна вычислительной сложности решения $p \cdot (c_0 + c_v)$ задач оценивания фазового состояния линейной системы ОДУ (p – число локаций исходного автомата H, c_0 – число входных локаций H, c_v – число циклов, в которых участвуют локации). Полученный автомат T_H имеет $p \cdot k(v)$ локаций ($k(v)$ – число различных областей начальных значений вектора переменных в локациях).

Для решения задач оценивания фазового состояния (функция "Cauchy" в алгоритме обобщенного таймер-преобразования) предлагается метод покоординатных оценок, который подробно рассмотрен в пункте 2.4. Он дает аппроксимацию множества достижимости линейной системы ОДУ с прямоугольной областью начальных значений, соответствующего истинности предиката (на дуге автомата или в локации), классом прямоугольных параллелепипедов в R^n, и оценки соответствующего этому множеству временного интервала.

В пункте 2.5 предложен метод оценки временного интервала в качестве частного решения задачи Коши (функция "Cauchy1" в алгоритме обобщенного таймер-преобразования).

2.2.2 Описание алгоритма

Алгоритм обобщенного таймер-преобразования можно представить в виде трех последовательных блоков:

Блок 1. Вычисление всех возможных множеств входных значений для каждой локации автомата и временных ограничений на всех дугах автомата, проверка на неразрастание помеченных циклов. Для этого находим локации автомата, на которых заданы области вход-

ных значений (строим множество E_0), а затем для каждого элемента множества E_0 строим дерево траекторий – дерево, содержащее все пути автомата, ведущие к следующим элементам множества E_0. При обнаружении несинхронизированного цикла, его начало добавляем в множество E_0, а конец помечаем. В процессе построения дерева траекторий помеченные элементы проверяются на неразрастание областей входных значений. В результате работы этой части алгоритма порождаются множества Q_i, содержащие для каждой локации автомата i все возможные области входных значений, дуги, на которых эти области были вычислены и времена переходов по этим дугам (в виде интервалов).

Блок 2. Разбиение локаций автомата на подлокации с наследованием дуг. В результате работы этого блока из множеств $\{V, E\}$ исходного гибридного автомата строим множество вершин и дуг $\{V_{new}, E_{new}\}$ нового автомата H_{new}.

Блок 3. Замена вектора фазовых переменных автомата на таймер, замена всех предикатов автомата на вычисленные временные ограничения в автомате H_{new}.

Опишем блоки алгоритма в стиле последовательного уточнения.

Блок 1. Построение множеств $\mathbf{Q}_1 \ldots \mathbf{Q}_p$.

```
const I*; // символ неизвестного интервала
const ∅; // пустое множество
type интервал : record {
a : real;
b : real;
};
type прямоугольная область = array [n] of интервал;
type предикат : string;
type дуга : record {
v : int; // локация-источник
v' : int; // целевая локация
pre : предикат; // условие перехода
post : предикат; // инициализация на дуге
};
type локация : record {
v : int; // номер локации
```

```
flow : string; // локальное поведение
inv : предикат; // инвариант локации
};
type элемент : record {
e : дуга;
I_e : интервал времени;
X_e : прямоугольная область;
};
type узел дерева : record {
e : дуга;
x : прямоугольная область;
};
var p : int; // количество локаций автомата
var n : int; // размерность автомата
var E, E_new : sets of дуга; // множества дуг исходного и результирующего автомата
var V,V_new : sets of локация; // множества локаций исходного и результирующего автомата
var v_egs : list of int; //список номеров посещаемых локаций
var Q_1,..., Q_p : sets of элемент;
var E_0 : set of узел дерева;
var E_c : set of узел дерева;
var e : дуга; // локальная переменная
var z : узел дерева; // локальная переменная
var q : элемент; // локальная переменная
var p1,p2 : предикат; // локальные переменные
var X : прямоугольная область; // локальная переменная
var v,v' : int; // локальные переменные - номера локаций

<инициализация множества E_0>;
<инициализация множеств Q_1,..., Q_p>;
M1:     <взять очередной элемент z из E_0>;
      v_egs := 0; E_c := 0;
      <построить дерево траекторий с корнем z>;
      if <не все элементы E_0 просмотрены> then goto M1;
      endif;
return (Q_1 ... Q_p).
```

```
<инициализация множества E0> :
  M2:     <взять очередную дугу e множества E>;
        if  e=Synch[X] or e=Init[X] then
              <создать новый элемент z в E0>;
              z.e:=e; z.x:=X;
        endif;
        if <не все дуги просмотрены> then goto M2;
        endif.
<инициализация множеств Q1, ..., Qk> :
  for  i=1...p
              Qi := ∅;
  endfor;
  M3:     <взять очередной элемент z множества E0>
        if   z.e=(v, v', p1, p2)  then
              <создать новый элемент q в Qv'>;
              q.e:=z.e;  q.Xe:=z.x;
              q.Ie:=I*; //значение интервала неизвестно
        endif;
        if <не все элементы просмотрены> then goto M3;
        endif.
<взять очередной элемент z из E0>:
  z:=nextnode(E0).
<создать новый элемент z в E0> :
  z:=newnode(); addnode(z, E0).
<создать новый элемент q в множестве Qi> :
  q:=newelement(); addelement(q, Qi).
<построить дерево траекторий с корнем z> :
  tree ( z : узел дерева ).
```

Блок 2. Разбиение на подлокации

```
  var V* : sets of int; // вспомогательное множество номеров локаций
  var D : sets of прямоугольная область; // вспомогательное множество областей входных значений подлокаций
  var vz : локация; // локальная переменная
  var s : int; // локальная переменная- номер локации
  var i,k : int; // локальные переменные

  i:=0; // инициализируем счетчик локаций в новом автомате
```

```
M0 :    <взять очередную локацию v из V>;
        D := ∅; // инициализируем множество областей входных значений локации v
        V* := ∅; // инициализируем множество номеров подлокаций локации v
M1:     <взять очередной элемент z множества Q_v>;
        if z.X_e ∉ v.inv then goto M1; // нач.множество не удовлетворяет инварианту локации v
        endif;
        if z.X_e = D[k] then // нач.множество уже встречалось
              s := V*[k]; // объединяем новую подлокацию с V*[k]
        else // создаем новую подлокацию локации v
              <создать новую локацию v_z>;
              v_z.v := i; v_z.inv := v.inv; v_z := v.flow;
              i := i+1;
              <добавить v_z в множество V_new >;
              <добавить v_z.v в множество V*>;
              s := v_z.v;
        endif;
M2:           <взять очередную дугу e из множества дуг E>;
              if e.v' = v and e∈ Q_i then
              <добавить e=(e.v',v_z,e.pre:=z.I_e,e.post:=z.X_e) в множество E_new >; // соединяем новую подлокацию с локацией, по которой шло вычисление
              <решить частный случай задачи Коши для входной области z.X_e и инварианта подлокации v_z, результат записать в v_z.inv>; // замена инварианта на временное ограничение
              endif;
              if <не все дуги рассмотрены> then goto M2;
        endif;
        if <не все элементы Q_v рассмотрены> then goto M1;
        endif;
        if <не все локации рассмотрены> then goto M0;
  endif;
  return ( V_new, E_new).
<взять очередную локацию v из V>:
  v := nextloc(V).
<создать новую локацию v_z>:
```

```
  v_z := newloc().
<добавить новую локацию v_z в множество V_new>:
  addloc(v_z,V_new).
<взять очередной элемент z множества Q_i>:
  z := nextelement(Q_i).
<взять очередную дугу e из E>:
  e := nextegs(E).
<добавить дугу e в множество E_new>:
  addegs(e,E_new).
<решить частный случай задачи Коши для входной области z.X_e и
инварианта подлокации v>:
  v_z.inv := Cauchy1 ( v , z.X_e, 0 ).
```

Блок 3. Замена вектора переменных на таймер.

```
  M0: <взять очередную локацию v множества V_new>;
        v.flow := "ẋ_x = 1"; // инвариант был заменен в блоке 2
  if <не все локации рассмотрены> then goto M0;
  endif;
  M1: <взять очередную дугу e множества E_new>;
        e.post := "t_x := 0"; // условие перехода было заменено в блоке 2
  if <не все дуги рассмотрены> then goto M1;
  endif;
  return ( V_new, E_new).
```

`func tree ( z : узел дерева );` // Рекурсивная функция построения дерева траекторий. Аргумент – переменная типа узел дерева, содержащий дугу z.e=(v,v'). Узлы дерева имеют тип "узел дерева", листьями дерева являются узлы, которые принадлежат множеству E_0, или узлы, которые принадлежат множеству E_c. Дерево строится методом поиска в глубину с одновременным построением множеств Q_i для всех посещенных локаций

```
  var I: интервал; // локальная переменная
  var x: прямоугольная область; //локальная переменная
  var outside : boolean; // локальная переменная
  var outres : boolean; // флаг неограниченности автомата
  var v : int; // номер текущей локации
  v := z.e.v';
```

```
M4:    <взять очередную дугу e=(v, v'', p1, p2), выходящую
из v>
       if ( ∃ q∈ Q_{v''} : q.e = e' ∧ q.I_e=I_*) then // до листа один
шаг
            <решить частный случай задачи Коши для локации
v с начальными условиями z.x и дуги e, результат записать в
переменную I>;
            q.I_e:=I;
       else
            <решить задачу Коши для локации v с начальными
условиями z.x и дуги e, результат записать в переменные I,
x и outside>;
            if outside then // область x неограничена на дуге e
                 <проверить свойство ограниченности гибрид-
ного автомата на дуге e>;
                 if <гибридный автомат неограничен>
                 return(fail - построение невозможно);
                 endif;
            endif;
            <создать новый элемент q множества Q_{v''}>;
            q.e:=e;  q.X_e:=x;  q.I_e:=I;
            if  v∉v_egs  then
                 <поместить v в список v_egs>;
            endif;
       endif;
       if  e∈ E_0 and e∉ E_c  then
       // дошли до следующей дуги из множества E_0 (лист дерева)
            return;
       endif;
       if  e∈ E_0 and e∈ E_c  then
       // обошли несинхронизированный цикл (лист дерева)
            <проверка цикла на неразрастание>;
            return;
       endif;
       if   v''∈v_egs  then
       // обнаружили несинхронизированный цикл с новой прямоугольной обла-
стью
```

```
                    <создать новый узел дерева z>;
                    z.e:=e;  z.x:=x;
                    <добавить z в множество E0>;
                    <добавить z в множество Ec>;
                    return;
        endif;
        tree(   z:= ( e, x )  );
        if <не все дуги рассмотрены> then goto M4;
        endif;
  return.
<решить задачу Коши для локации v с начальными условиями z.x
и дуги e>:
  {I,x,outside}:=Cauchy ( v , z.x, e).
<решить частный случай задачи Коши для локации v с начальными
условиями z.x и дуги e>:
  I := Cauchy1 ( v , z.x, e ).
<создать новый узел дерева z>:
  z := newnode().
<добавить z в множество E>:
  addnode(z,E).
<проверить свойства ограниченности гибридного автомата на ду-
ге e> :
  outres := analyze(x, e).
<проверка цикла на неразрастание> :
  <найти элемент z в Ec, такой что z.e=e>;
  if  x ⊈ z.x  then
        return ( fail -- построение невозможно);
        // цикл не удовлетворяет условию неразрастания
<гибридный автомат неограничен> :
  outres ≠ 0.
<поместить локацию v в список v_egs> :
  addlistloc( v, v_egs ).
<найти элемент z в Ec, такой что z.e=e>;
  z:= findnode( Ec, e ).
func analyze (x : прямоугольная область, e : дуга):boolean;
   // возвращает результат анализа гибридного автомата на ограниченность на дуге
e (true – неограничен, false – ограничен)
```

```
for k := 1 ... n
      if x[k] = Infty_ or x[k] = Infty+ then
            if e.post[x_k] ≠ "x'_k = [a, b]" ∧ [a, b] ∈ R
            or e.pre[x_k] ≠ "a ≤ x_k ≤ b" ∧ [a, b] ∈ R
            or e.post[x_k] ≠ "x'_k = x_i" ∧ x_i ≠ Infty then
// компонента вектора после инициализации осталась неограниченной
                  return true;
            endif;
      endif;
endfor;
return false.
```

---

func newnode () : узел дерева; // создает объект типа узел дерева

func nextnode (E : set of узел дерева) : узел дерева; // возвращает следующий элемент множества E

func addnode (z : узел дерева, E : set of узел дерева); //добавляет элемент z в множество E

func findnode (E: set of узел дерева, e : дуга) : узел дерева; // возвращает элемент множества E, содержащий заданную дугу

func newloc (); // создает объект типа локация

func nextloc (V : set of int) : int; // возвращает следующую локацию из множества локаций

func addloc (V : set of int, v : int); // добавляет локацию к множеству V

func nextegs (E : set of дуга) : дуга; // возвращает следующую дугу из множества E

func addegs (E : set of дуга, e : дуга); // добавляет дугу e к множеству E

func newelement (); //создает объект типа элемент

func addelement (q : элемент, Q : set of элемент); //добавляет элемент q в множество Q

func addlistloc (v : int; vlist : list of int) //добавляет локацию к списку локаций

func Cauchy (v : локация; x_0 : прямоугольная область; e: дуга):{I : интервал; x : прямоугольная область; out : boolean};

// аргументы: v – текущая локация, x_0 – область входных значений локации v, e –

дуга, выходящая из локации v. Возвращаемые значения: I – интервал времени возможного перехода по дуге e, x – область значений переменных на конце дуги e, out – флаг неограниченности вычисленной области x. Алгоритм представлен в п.2.4

`func Cauchy1 (v : локация; x`$_0$ `: прямоугольная область; e: дуга) : I : интервал;` //аргументы: v – текущая локация, x_0 – область входных значений локации v, e – дуга, выходящая из локации v, причем e = 0 или e$\in$E$_0$. Возвращаемое значение: если e $\in$ E$_0$, I – интервал времени возможного перехода по дуге e, если e = 0, I – интервал времени возможного нахождения в локации v. Алгоритм представлен в п.2.5

2.3 Теоремы об алгоритме обобщенного таймер-преобразования

Пусть H – гибридный автомат с линейными системами ОДУ, T_H – линейная аппроксимация гибридного автомата H, построенная с помощью обобщенного таймер-преобразования.

Теорема 2.3.1 *Автомат T_H, построенный для гибридного автомата H с помощью обобщенного таймер–преобразования, является системой временных переходов.*

Доказательство: Из схемы алгоритма следует, что автомат T_H имеет единственный таймер, который перезапускается во всех подлокациях, а условия переходов по дугам автомата T_H в соответствии со схемой алгоритма представляют собой временные ограничения, заданные в виде интервалов. Таким образом, в соответствии с определением 1.2.3, будем иметь систему временных переходов.□

Теорема 2.3.2 *Автомат T_H допускает все пробеги, допустимые в исходном автомате H. Обратное утверждение неверно:*

$$Reach(H) \subseteq Reach(T_H).$$

Для доказательства этой теоремы приведем два утверждения относительно оценок, полученных методом покоординатных оценок и методом оценки временного интервала (см. в п.2.4, 2.5):

Лемма 2.3.1 *Прямоугольный параллелепипед D, построенный методом покоординатных оценок, является оптимальной верхней оценкой области первого пересечения пучка решений задачи Коши и полупространства, заданного линейным предикатом, в классе прямоугольных параллелепипедов.*

Интервал времени $\mathcal{I}$, построенный методом покоординатных оценок, описывает точные временные границы, когда пучок решений задачи Коши в первый раз будет находиться в полупространстве, заданном линейным предикатом.

Лемма 2.3.2 *Интервал времени $\mathcal{I}$, построенный методом оценки временного интервала, является верхней оценкой точных временных границ, когда пучок решений задачи Коши в первый раз будет находиться в полупространстве, заданном линейным предикатом.*

Доказательство: Из утверждений 2.3.1, 2.3.2 следует, что метод покоординатных оценок и метод оценки временного интервала, используемые при решении задач оценивания фазового состояния в алгоритме обобщенного таймер-преобразования, дают верхнюю аппроксимацию множеств достижимости линейных систем ОДУ. Таким образом, для каждой вершины v гибридного автомата мы получаем оценку множеств достижимости вершины $Reach(v)_{[\text{оц.}]}$, которая обладает свойством $Reach(v) \subseteq Reach(v)_{[\text{оц.}]}$. Но если множества достижимости каждой вершины гибридного автомата обладают этим свойством, то, очевидно, что характеристическое множество $Reach(H)$ гибридного автомата H попадет в характеристическое множество $Reach(T_H)$ построенного автомата T_H. При этом множество $Reach(T_H)$ будет шире множества $Reach(H)$. □

Из теоремы 2.3.1 следует, что к линеаризованному автомату T_H может быть применен метод символьной верификации (или другие известные методы верификации систем временных переходов).

В свою очередь, из теоремы 2.3.2 следует, что если характеристическое множество автомата T_H будет удовлетворять верифицируемым требованиям к поведению, то и характеристическое множество исходного автомата также будет им удовлетворять, однако если если характеристическое множество автомата T_H не будет удовлетворять верифицируемым требованиям к поведению, о характеристическом

множестве исходного автомата ничего сказать нельзя. Таким образом, имеем следующее важное следствие:

Следствие 2.3.1 *Метод символьной верификации можно использовать для верификации тех свойств поведения гибридного автомата с линейными системами ОДУ, результат верификации которых положителен.*

Пусть теперь дана параллельная композиция из m гибридных автоматов, часть из которых является гибридными автоматами с линейными системами ОДУ, а часть принадлежит к классам гибридных автоматов, для которых доказана сходимость метода символьной верификации, например они являются таймированными автоматами (прямоугольными автоматами, линейными гибридными автоматами с постоянными коэффициентами). Предположим также, что множества фазовых переменных гибридных автоматов не пересекаются, но $\Sigma_i \cap \Sigma_j \neq \emptyset$, где Σ_i – алфавит событий автомата i.

Применим алгоритм обобщенного таймер-преобразования последовательно для всех гибридных автоматов с линейными системами ОДУ и заменим гибридные автоматы H_i с линейными системами ОДУ на соответствующие системы временных переходов T_i. Получим параллельную композицию L автоматов $T_1, \ldots, T_k, H_{k+1}, \ldots, H_m$.

Теорема 2.3.3 *Пусть L – параллельная композиция автоматов $H_1, \ldots, H_k, T_{k+1}, \ldots, T_m$, в которой $H_1, \ldots H_k$ – гибридные автоматы с линейными системами ОДУ, $T_{k+1}, \ldots, T_m$ – таймированные автоматы. Пусть множества фазовых переменных гибридных автоматов $H_1, \ldots H_k$ не пересекаются, то есть $\forall i, j,\ 1 \leq i \leq k,\ 1 \leq j \leq m,\ i \neq j,\ X_{H_i} \cap X_{H_j} = \emptyset,\ X_{H_i} \cap X_{T_j} = \emptyset$. Тогда автомат T_L, построенный с помощью обобщенного таймер–преобразования для параллельной композиции автоматов L, является таймированным автоматом.*

Доказательство: Если множества фазовых переменных гибридных автоматов с линейными системами ОДУ не пересекаются, то к каждому автомату можно применить обобщенное таймер-преобразование. Действительно, в этом случае предикаты на дугах и предикаты-инварианты не будут содержать переменных, значения

которых зависят от других автоматов, следовательно, возможно решение задачи оценивания фазового состояния для всех локаций автомата. А тогда это утверждение следует из теоремы 2.3.1, эквивалентности моделей системы временных переходов и модели таймированного автомата, доказанной в работе [28], и определения параллельной композиции таймированных автоматов 1.2.6. □

Аналогичным образом могут быть доказаны следующие теоремы.

Теорема 2.3.4 *Пусть L – параллельная композиция автоматов $H_1, \ldots, H_k, H_{k+1}, \ldots, H_m$, в которой $H_1, \ldots H_k$ – гибридные автоматы с линейными системами ОДУ, $H_{k+1}, \ldots, H_m$ – прямоугольные автоматы с ограниченной интервальной инициализацией. Пусть множества фазовых переменных гибридных автоматов $H_1, \ldots H_k$ не пересекаются, то есть $\forall i, j,\ 1 \leq i \leq k,\ 1 \leq j \leq m,\ i \neq j,\ X_{H_i} \cap X_{H_j} = \emptyset$. Тогда автомат T_L, построенный с помощью обобщенного таймер–преобразования для параллельной композиции автоматов L, является прямоугольным автоматом с ограниченной интервальной инициализацией.*

Теорема 2.3.5 *Пусть L – параллельная композиция автоматов $H_1, \ldots, H_k, H_{k+1}, \ldots, H_m$, в которой $H_1, \ldots H_k$ – гибридные автоматы с линейными системами ОДУ, $H_{k+1}, \ldots, H_m$ – линейные гибридные автоматы с постоянными коэффициентами. Пусть множества фазовых переменных гибридных автоматов $H_1, \ldots H_k$ не пересекаются, то есть $\forall i, j,\ 1 \leq i \leq k,\ 1 \leq j \leq m,\ i \neq j,\ X_{H_i} \cap X_{H_j} = \emptyset$. Тогда автомат T_L, построенный с помощью обобщенного таймер–преобразования для параллельной композиции автоматов L, является линейным гибридным автоматом с постоянными коэффициентами.*

Из теорем 2.3.3, 2.3.4, 2.3.5 следует, что метод символьной верификации применим к параллельной композиции гибридных автоматов с линейными системами ОДУ.

Из теоремы 2.3.2 следует, что если характеристическое множество автомата T_L будет удовлетворять верифицируемым требованиям к поведению, то и характеристическое множество исходной параллельной композиции также будет им удовлетворять, однако если если

характеристическое множество автомата T_L не будет удовлетворять верифицируемым требованиям к поведению, о характеристическом множестве параллельной композиции L ничего сказать нельзя. Таким образом, имеем следующее важное следствие:

Следствие 2.3.2 *Метод символьной верификации можно использовать для верификации тех свойств поведения параллельной композиции гибридных автоматов с линейными системами ОДУ, результат верификации которых положителен.*

Сформулируем основные теоремы о применимости алгоритма обобщенного таймер-преобразования к гибридному автомату с линейными системами ОДУ.

Теорема 2.3.6 *Обобщенное таймер–преобразование применимо к гибридному автомату с линейными системами дифференциальных уравнений, если количество возможных интервально заданных начальных значений вектора переменных в каждой локации автомата конечно.*

Доказательство: Метод символьной верификации применим к моделям с конечным числом состояний. Количество подлокаций, на которые разбивается каждая локация гибридного автомата на шаге 1 алгоритма обобщенного таймер-преобразования, равно количеству различных множеств начальных значений вектора фазовых переменных в этой локации. Очевидно поэтому, что конечность построенного автомата и терминальность алгоритма, определяется конечным числом возможных интервально заданных начальных значений переменных в каждой локации. □

Теорема 2.3.7 *Количество возможных интервально заданных начальных значений вектора переменных* $\mathbf{x}$ *в локации* v *гибридного автомата с линейными системами ОДУ конечно, если*

1) в предикате init автомата имеется инициализация всех переменных в виде ограниченных интервалов;

2) каждый цикл в автомате, проходящий через локацию v*,*

- *либо является синхронизированным, то есть можно найти дугу e, для которой выполнено одно из следующих условий:*

  — имеется инициализация всех переменных одновременно в виде ограниченных интервалов ("синхронизированный вход")

  $$post(e) : \{x_i := c_i,\ i = 1 \dots n\},\ c_i \in [l_i, u_i],\quad [l_i, u_i] \in \mathcal{R}$$

  — условие перехода по дуге содержит ограничения на все переменные одновременно в виде ограниченных интервалов ("синхронизированный выход")

  $$pre(e) : \wedge_{i=1}^{n} l_i \le x_i \le u_i,\quad [l_i, u_i] \in \mathcal{R}$$

- *либо обладает свойством неразрастания фазовых переменных, то есть при обходе цикла все возможные области начальных значений $\{\mathbf{x}_0(v)\}$ вершины v не расширяются:*

  $$\forall\{\mathbf{x}_0(v)\} : \{\mathbf{x}_0(v)\}_{\text{после обхода}} \subseteq \{\mathbf{x}_0(v)\}_{\text{до обхода}}$$

Доказательство: Следуя обратно по любому пути, ведущему в локацию v, при данных условиях теоремы мы через конечное число шагов придем к месту, где будут известны значения компонент вектора переменных в виде конечных интервалов (либо на входе, либо на выходе некоторой локации) или вернемся по некоторому циклу в локацию v. В первом случае, зная цепочку систем дифференциальных уравнений и область начальных значений, возможно проследить развитие вектора переменных по каждому пути вплоть до локации v (путем последовательного решения задач оценивания фазового состояния линейной системы ОДУ, см. функцию "Cauchy"). Во втором случае возможно проследить развитие вектора переменных при обходе цикла со всеми вычисленными для первого случая интервальными входными значениями. Исследование на разрастание областей входных значений в несинхронизированных циклах может быть также сведено к последовательности решений задач оценивания.

Так как число начальных локаций и число циклов, проходящих через произвольную вершину конечного автомата, конечно, то для первого случая получим конечное число интервальных векторов, описывающих области начальных значений вектора переменных системы в локации. При условии неразрастания областей входных значений

локации v при обходе по несинхронизированному циклу количество интервально заданных входных значений вершины v не изменится. □

Теорема 2.3.8 *Обобщенное таймер–преобразование применимо к гибридному автомату с линейными системами дифференциальных уравнений, если он обладает свойством ограниченной интервальной инициализации и если в процессе построения не нарушается свойство ограниченности областей входных значений каждой локации автомата.*

Доказательство: Области начальных значений при решении последовательности задач оценивания фазового состояния при построении деревьев траекторий на шаге 1 алгоритма обобщенного таймер-преобразования не могут иметь неограниченные компоненты. □

2.4 Метод покоординатных оценок. Функция "Cauchy"

2.4.1 Постановка задачи и обоснование решения

Сформулируем задачу оценивания фазового состояния линейной системы ОДУ при переходе по заданной дуге гибридного автомата.

Пусть дана линейная система дифференциальных уравнений с постоянными коэффициентами размерности n:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + \mathbf{b}; \quad a_{ij}, b_i \in R, \quad i, j = 1 \dots n \tag{11}$$

Область начальных значений задана прямоугольным параллелепипедом в R^n:

$$\mathbf{x}_0 \in D_0, \quad D_0 \in \mathcal{P} \tag{12}$$

Требуется
1) вычислить нижнюю и верхнюю временные границы t_1, t_2, когда компоненты пучка решений задачи Коши (11)-(12) $\mathbf{x}(t)$ будут удовлетворять линейному предикату

$$P \ : \quad \Sigma_{i=1}^{n} h_i x_i \sim c, \quad h_i, c \in R, \quad \sim = \{<, \leq, =, \geq, >\} \tag{13}$$

то есть когда пучок решений задачи Коши (11)-(12) в первый раз будет находиться в области фазового пространства системы, ограниченной неравенством $\Sigma_{i=1}^{n} h_i x_i \sim c$ *(полупространство).*
2) оценить множество значений вектора $\mathbf{x}$ *(прямоугольным параллелепипедом в* R^n*), соответствующее первому пересечению пучка решений задачи Коши (11)–(12)* $\mathbf{x}(t)$ *и соответствующей области фазового пространства системы.*

Как известно, множеством достижимости $D(t, s, M)$ системы ОДУ, с начальными условиями вида $\mathbf{x}(s) \in M$, где s – начальный момент времени, $M \subset R^n$ – множество начальных состояний системы, называется совокупность концов всех траекторий $\mathbf{x}(t)$ системы ОДУ, начинающихся в момент s в точках начального множества M.

Объединение множеств достижимости $D(t, s, M)$ при всех $t \geq s$ называется интегральной воронкой системы ОДУ. Интегральная воронка состоит из всех точек фазового пространства, которые могут быть достигнуты в какой-либо момент времени[8].

Докажем важное утверждение о множестве достижимости линейной системы ОДУ вида (11)–(12).

Теорема 2.4.1 *В каждый фиксированный момент времени множество достижимости линейной системы ОДУ с постоянными коэффициентами и множеством начальных значений, заданных прямоугольным параллелепипедом в* R^n*, представляет собой (косоугольный) параллелепипед в* R^n*.*

Доказательство: Фундаментальная матрица линейной системы ОДУ с постоянными коэффициентами представляет собой матричную экспоненту $X(t) = e^{At}$.

Согласно теореме о о существовании и единственности решений линейной системы ОДУ [6], решение однородной системы существует при всех $t \geq s$, является единственным и может быть записано в виде

$$\mathbf{x} = X(t)\ \mathbf{c},$$

[8]Понятие интегральной воронки эквивалентно понятию глобального множества достижимости гибридной системы *Reach*, нахождение которого составляет основную часть метода символьной верификации.

где $X(t)$ – фундаментальная матрица системы, $\mathbf{c}$ – вектор начальных значений.

Решение неоднородной системы ОДУ

$$\frac{d\mathbf{x}}{dt} = A(t)\mathbf{x} + \mathbf{f}(t), \quad \mathbf{x}(s) = \mathbf{c}, \quad t \geq s \tag{14}$$

может быть записано соответственно в виде:

$$\mathbf{x} = X(t)\ \mathbf{c} + \int_0^t X(t)X^{-1}(\tau)f(\tau)d\tau.$$

Для линейной системы дифференциальных уравнений с постоянными коэффициентами, таким образом, будем иметь:

$$\mathbf{x}(t) = e^{At}\mathbf{c}$$

для однородной системы и

$$\mathbf{x}(t) = e^{At}\mathbf{c} + \textstyle\int_s^t e^{A(t-\tau)}\mathbf{f}(\tau)d\tau$$

для неоднородной системы.

Если $\mathbf{f} = \mathbf{b}$ – постоянный вектор, то:

$$\mathbf{x}(t) = e^{At}\mathbf{c} + A^{-1}(e^{At} - I)\mathbf{b}$$

(если A^{-1} существует).

Зафиксируем момент времени $h \geq 0$ и получим выражение для $\mathbf{x}(t+h)$, используя $\mathbf{x}(t)$. В случае однородного векторного уравнения в силу свойств матричной экспоненты получим:

$$\mathbf{x}(t+h) = e^{A(t+h)}\mathbf{c} = e^{Ah}e^{At}\mathbf{c} = e^{Ah}\mathbf{x}(t).$$

Для неоднородного уравнения будем иметь:

$$\mathbf{x}(t+h) = e^{A(t+h)}\mathbf{c} + A^{-1}e^{A(t+h)}\mathbf{b} - A^{-1}\mathbf{b} = e^{At}e^{Ah}\mathbf{c} + A^{-1}e^{At}e^{Ah}\mathbf{b} - A^{-1}\mathbf{b}$$

$$\mathbf{x}(t+h) = e^{Ah}\mathbf{x}(t) + A^{-1}(e^{Ah} - I)\mathbf{b}.$$

Поскольку при фиксированном времени и постоянной матрице A матричная экспонента e^{Ah} – некоторая постоянная невырожденная матрица и $A^{-1}(e^{Ah} - I)$ – также постоянная матрица, то в каждый фиксированный момент времени будем иметь линейное преобразование вектора $\mathbf{x}$ как для однородной, так и для неоднородной системы ОДУ (в последнем случае добавляется сдвиг координат). Любое линейное невырожденное преобразование отображает заданный в R^n прямоугольный параллелепипед в некоторый косоугольный параллелепипед в R^n, поскольку класс параллелепипедов инвариантен относительно афинных преобразований.

Поэтому, задав в качестве t начальный момент времени s в формулах для $\mathbf{x}(t+h)$, в силу единственности фундаментальной матрицы системы ОДУ, получим требуемый результат. □

2.4.2 Описание метода покоординатных оценок

Утверждение 2.4.1 дает нам право свести задачу оценки области пересечения интегральной воронки линейной системы ОДУ и полупространства (или гиперплоскости), заданного предикатом (13), к исследованию решений задач Коши, с начальными условиями в вершинах заданного прямоугольного параллелепипеда, в заданном полупространстве. Назовем этот метод методом покоординатных оценок.

Введем два вектора $\mathbf{x}_{max}, \mathbf{x}_{min}$ – для хранения экстремальных точек пучка решения задачи Коши (11)-(12) в заданном полупространстве, и две вещественные переменные t_{min}, t_{max} – для хранения границ искомого интервала времени.

Найдем последовательно экстремальные значения $\mathbf{x}^{(1)}_{min}, \mathbf{x}^{(1)}_{max}, \ldots, \mathbf{x}^{(m)}_{min}, \mathbf{x}^{(m)}_{max}$ граничных решений задачи Коши (11)-(12) и соответствующие временные интервалы $t^{(1)}_{min}, t^{(1)}_{max}, \ldots, t^{(m)}_{min}, t^{(m)}_{max}$ ($m = 2^n$, функция "exstrems"), и выберем из них минимальное и максимальное значения

$$\mathbf{x}_{min} = min\{\mathbf{x}^{(1)}_{min}, \ldots, \mathbf{x}^{(m)}_{min}\}, \quad \mathbf{x}_{max} = max\{\mathbf{x}^{(1)}_{max}, \ldots, \mathbf{x}^{(m)}_{max}\}$$

$$t_{min} = min\{t^{(1)}_{min}, \ldots, t^{(m)}_{min}\}, \quad t_{max} = max\{t^{(1)}_{max}, \ldots, t^{(m)}_{max}\} \qquad (15)$$

Если после перебора всех граничных решений окажется, что $\mathbf{x}_{min} = \mathbf{x}_{max} = \emptyset$, то это значит, что пучок решений задачи Коши никогда не окажется в заданном полупространстве. В алгоритме обобщенного таймер-преобразования соответствующая дуга должна быть удалена (предикат P заменен на $False$).

Если после перебора окажется, что $t_{min} = 0, t_{max} = \infty$ то это значит, что пучок решений задачи Коши всегда будет находиться в заданном полупространстве. В обобщенном таймер-преобразовании предикат P на соответствующей дуге должен быть заменен на $True$.

В остальных случаях получим прямоугольную область $D = [\mathbf{x}_{min}, \mathbf{x}_{max}]$ (возможно неограниченную) и временной интервал $\mathcal{I} = [t_{min}, t_{max}]$.

Прямоугольная область D окажется неограниченной, если хотя бы одна компонента вектора $\mathbf{x}_{min}$ ($\mathbf{x}_{max}$) будет равна $-\infty(+\infty)$. При обнаружении этого факта в алгоритм обобщенного таймер-преобразования передается флаг и проверяется сохранение свойства ограниченной интервальной инициализации гибридного автомата (функция "analyze" в п.2.2.2).

Вычислительная сложность метода покоординатных оценок складывается из вычислительной сложности решения 2^n точечных задач Коши (вызова функции "extrems", которая описана в следующем пункте).

Опишем метод покоординатных оценок в стиле последовательного уточнения. Аргументами функции "Cauchy" являются текущая локация, область входных значений локации и дуга, выходящая из локации, для которой вычисляется время перехода и область выходных значений для дуги. Возвращаемые значения: I – интервал времени возможного перехода по заданной дуге, D – область значений переменных на конце дуги e, *out* – флаг ограниченности вычисленной области x.

Функция "Cauchy". Метод покоординатных оценок.

```
const Infty+, Infty-; // обозначение бесконечных значений
const ∅; // пустое множество
type Real= set of {reals ∪ Infty+ ∪ Infty-};
type интервал времени : record {
t_min : Real;
t_max : Real;
};
type прямоугольная область = array [n] of интервал;
type предикат : string;
type дуга : record {
v : int; // локация-источник
v' : int; // целевая локация
pre : предикат; // условие перехода
post : предикат; // инициализация на дуге
};
var n : int; // размерность автомата
var x_min : vector[n] of Real; // вектор нижнего экстремального значения x(t)
var x_max : vector[n] of Real; // вектор верхнего экстремального значения x(t)

func Cauchy (v : локация; D_0 : прямоугольная область; e : дуга):{I : интервал; D : прямоугольная область; out : boolean};
var x_m, x_M : vector[n] of Real; // локальные переменные
```

Рис. 9: Блок-схема метода покоординатных оценок

```
  var t1, t2 : Real; // локальные переменные
  var P : предикат;
        outside := false;
        xmax = ∅; xmin = ∅; // инициализация экстремальных значений
        tmin = ∅; tmax = ∅; // инициализация временного интервала
        P := e.pre;
  M1 : <выбрать очередную вершину x0 прямоугольной области D0>;
        <найти экстремальные значения задачи Коши dx/dt = Ax +
b,
x(0) = x0 в области P, результат записать в xm,xM,t1,t2>;
              xmin = min {xmin, xm};
              xmax = max {xmax, xM};
              tmin = min {tmin, t1};
              tmax = max {tmax, t2};
              if xmin = Infty− or xmax= Infty+ then // прямоуголь-
ная область неограничена
                    outside:=true;
              endif;
              if <не все вершины D0 перебраны> then goto M1;
              endif;
        if xmin=xmax = ∅ then   return ("never"); // x(t) никогда
не попадет в область P
        endif;
        if tmin=0, tmax = ∞     return ("always"); // x(t) все-
гда будет находиться в области P
        endif;
        return (D :=[ xmin, xmax ], I := [ tmin, tmax ] ).
<выбрать очередную вершину x0 прямоугольной области D0>:
  x0 := nextvertex( D0 ).
<найти экстремальные значения задачи Коши dx/dt = Ax + b, x(0) =
x0 в области P, результат записать в xm, xM, t1, t2 > :
  { xm, xM, t1, t2 } := extrems ( A, b, P, x0 ).
```

```
func nextvertex( D0 : прямоугольная область) : вектор; // воз-
вращает следующую вершину заданного прямоугольного параллелепипеда
func extrems (A: matrix[n,n] of real, b: vector[n] of real,
P : предикат, x0 : прямоугольная область) : {X1, X2 : вектор,
```

`T1, T2 : Real};` // находит пересечение точечной задачи Коши с заданной областью P. Возвращаемые значения: минимальное и максимальное значение решения задачи Коши в области P, границы временного интервала пересечения решения и области. Алгоритм представлен в п.2.4.3

2.4.3 Функция "extrems" для метода покоординатных оценок

Пусть дана линейная система дифференциальных уравнений с постоянными коэффициентами размерности n:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + \mathbf{b}; \quad \mathbf{x}(0) = \mathbf{x}_0,\ a_{ij}, b_i \in R, \tag{16}$$

Ставится задача поиска экстремальных точек решения (16) $\mathbf{x}(t)$, удовлетворяющих предикату

$$P\ :\quad \Sigma_{i=1}^{n} h_i x_i \sim c,\ h_i, c \in R,\ \sim = \{<, \leq, =, \geq, >\} \tag{17}$$

(поиска экстремальных точек $\mathbf{x}(t)$ в области, заданной неравенством $\Sigma_{i=1}^{n} h_i x_i \sim c$) и вычисления соответствующего интервала времени.

Опишем основные шаги алгоритма решения.

Шаг 1. Предположим, что мы нашли невырожденное преобразование координат S, приводящее матрицу A к квазидиагональному виду Λ. Если система дифференциальных уравнений (16) однородна, то получим решение задачи Коши в новых координатах:

$$\hat{\mathbf{x}} = e^{\Lambda t}\hat{\mathbf{x}}_0 \tag{18}$$

где $\hat{\mathbf{x}}$ – вектор переменных системы ОДУ в новом базисе ($\hat{\mathbf{x}} = S^{-1}\mathbf{x}$), $\hat{\mathbf{x}}_0$ – начальная точка в новом базисе ($\hat{\mathbf{x}}_0 = S^{-1}\mathbf{x}_0$), Λ – матрица, которая в общем случае составлена из s жордановых ящиков, соответствующих различным собственным значениям $\lambda_i,\ i = 1 \ldots s$ кратности m_i.

Для неоднородной системы ОДУ будем иметь (если A^{-1} существует):

$$\hat{\mathbf{x}}(t) = e^{\Lambda t}\hat{\mathbf{x}}_0 + A^{-1}(e^{\Lambda t} - I)\hat{\mathbf{b}}, \tag{19}$$

где $\hat{\mathbf{b}} = S^{-1}\mathbf{b}$.

Если среди собственных значений матрицы A найдется хотя бы одна комплексно–сопряженная пара $\lambda_{i,i+1}$, то функции $\hat{x}_i(t)$ будут комплексно-значные ($s_{ij},\ \hat{b}_i,\ \hat{x}_{i0}\ \in C$).

Шаг 2. Предикат (17) в новом базисе будет описываться следующим очевидным образом (при замене переменных x_i на $\sum_{j=1}^{n} s_{ij}\hat{x}_j$):

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_i s_{ij}\hat{x}_j \le c = \sum_{j=1}^{n}\hat{h}_j\hat{x}_j \le c,$$

где $\hat{h}_j = \sum_{i=1}^{n} h_i s_{ij}$. В векторном виде

$$\hat{P} : \hat{\mathbf{h}}\hat{\mathbf{x}}\ \le\ c, \quad \hat{\mathbf{h}} = \mathbf{h}^T S \tag{20}$$

Шаг 3. Овеществим предикат (20) и копомпонентные решения $\hat{x}_i(t)$. Будем иметь следующие основные случаи:

1) при попарно-различных вещественных собственных значениях $\lambda_1, \ldots, \lambda_n$ матрицы A покомпонентные решения имеют вид:

$$\hat{x}_i(t) = \hat{x}_{i0}e^{\lambda_i t},\ i = 1 \ldots n;$$

для однородной системы ОДУ,

$$\hat{x}_i(t) = -\frac{\hat{b}_i}{\lambda_i} + e^{\lambda_i t}(\frac{\hat{b}_i}{\lambda_i} + \hat{x}_{i0})$$

для неоднородной системы ОДУ и $\lambda_i \neq 0$ (для $\lambda_i = 0\ \ \hat{x}_i = \hat{x}_{i0}$).

2) при существовании среди собственных значений матрицы A вещественного собственного значения λ_k кратности $m_k > 1$ покомпонентные решения $\hat{x}_k, \ldots \hat{x}_{(k+m_k)}$ имеют вид:

$$\hat{x}_j(t) = P_{l-1}(t)e^{\lambda_k t},\ j = k \ldots (k + m_k),$$

для однородной системы ОДУ, где $P_{l-1}(t)$ – полином степени $l = k + m_k - j$ с коэффициентами $\hat{x}_{j0}, \ldots, \frac{\hat{x}_{l0}}{(l-1)!}$, или

$$\hat{x}_j(t) = P_{l-1}(t)e^{\lambda_k t} - \frac{\tilde{b}_j}{\lambda_k},\ j = k \ldots (k + m_k),\quad \lambda_k \neq 0$$

для неоднородной системы ОДУ, где $P_{l-1}(t)$ – полином вида $\tilde{x}_{jo} + \tilde{x}_{(j+1)0}t + \ldots + \frac{t^{l-1}}{(l-1)!}\tilde{x}_{l0}$ степени $l = i + m_k - j$, а $\tilde{b}_j$ и коэффициенты

полинома $\tilde{x}_{j0}, \dots \tilde{x}_{l0}$ могут быть вычислены по рекуррентным формулам:

$$\tilde{b}_{(i+m_k)} = \hat{b}_{(i+m_k)}, \quad \tilde{b}_s = \hat{b}_s - \frac{1}{\lambda_k}\tilde{b}_{s+1}, \quad s = j \dots i + m_k - 1;$$

$$\tilde{x}_{(i+m_k)0} = \hat{x}_{(i+m_k)0}, \quad \tilde{x}_{s0} = \hat{x}_{s0} + \frac{1}{\lambda_k}\tilde{b}_s, \quad s = j \dots i + m_k - 1.$$

3) при существовании среди собственных значений матрицы A комплексно-сопряженной пары $\lambda_{i,i+1} = \alpha^{(i)} \pm i\beta^{(i)}$ $(\beta^{(i)} \neq 0)$ кратности 1 соответствующие покомпонентные решения $\hat{x}_i, \hat{x}_{i+1}$ имеют вид:

$$\hat{x}_i(t) = Re\{\hat{x}_{i0}\}\ e^{\alpha^{(i)}t}cos(\beta^{(i)}t) - Im\{\hat{x}_{i0}\}e^{\alpha^{(i)}t}sin(\beta^{(i)}t)$$

$$\hat{x}_{i+1}(t) = Re\{\hat{x}_{(i+1)0}\}\ e^{\alpha^{(i)}t}cos(\beta^{(i)}t) + Im\{\hat{x}_{(i+1)0}\}e^{\alpha^{(i)}t}sin(\beta^{(i)}t)$$

для однородной системы ОДУ, Re и Im – вещественная и мнимая части комплексного числа, или

$$\hat{x}_i(t) = Re\{\frac{1}{abs(\alpha+i\beta)^4}\{abs(\alpha+i\beta)^2(-\beta c-\alpha d+(\alpha^2+\beta^2)e^{\alpha t}\hat{x}_0cos(\beta t))-$$

$$(\alpha^2+\beta^2)e^{\alpha t}(-(\beta c+\alpha d)cos(\beta t)+(\alpha c-\beta d)sin(\beta t))\}\}$$

$$\hat{x}_{i+1}(t) = Re\{\frac{1}{abs(\alpha-i\beta)^4}\{abs(\alpha-i\beta)^2(\beta c-\alpha d+(\alpha^2+\beta^2)e^{\alpha t}\hat{x}_0cos(\beta t))+$$

$$(\alpha^2+\beta^2)e^{\alpha t}(-\beta c+\alpha d)cos(\beta t)+(\alpha c+\beta d)sin(\beta t))\}\}$$

для неоднородной системы ОДУ, $\hat{b}_i = d + ic$.

4) при существовании среди собственных значений матрицы A комплексно-сопряженной пары $\lambda_{i,i+1} = \alpha \pm i\beta$ кратности $m_k > 1$ $(\beta^{(i)} \neq 0)$ соответствующие компоненты $x_i \dots x_{i+2m_k}$ имеют вид квазимногочленов:

$$\hat{x}_j = P_{l-1}(t)e^{\alpha t}cos(\beta t)\ +\ Q_{l-1}(t)e^{\alpha t}sin(\beta t),\ j = i \dots 2m_k,\ l = i + m_k - j$$

где $P_{l-1}(t), Q_{l-1}(t)$ – полиномы степени не большей $m_k - 1$ с коэффициентами $\tilde{x}_{i0}, \dots, \tilde{x}_{(i+2m_k)0}$, которые аналогично предыдущим случаям однозначно вычисляются из начальных условий.

Шаг 4. Зная покомпонентные решения задачи Коши в явном виде в новом базисе $\hat{x}_i(t)$, можно определить экстремальные значения каждой компоненты вектора фазовых переменных в полупространстве, заданном предикатом $\hat{P}$ (в новом базисе). Для этого

1) составляем вещественную функцию одной переменной

$f(t) = \sum_{i=1}^{n} \hat{h}_i \hat{x}_i(t)$, где $\hat{x}_i(t)$ – покомпонентные овеществленные решения задачи Коши, полученные на шаге 3, $\hat{h}_i$ – коэффициенты неравенства (20);

2)находим два первые неотрицательные корня t_1, t_2 уравнения $f(t) - c = 0$;

3) исследуем вещественные функции $\hat{x}_i(t)$ на экстремумы в интервале $[0, t_1]$ или $[t_1, t_2]$ в зависимости от принадлежности начальной точки $\hat{x}_{i0}$ к полупространству $\hat{P}$ (подробно см. описание функции "exstrems").

В результате получим два вектора $\hat{\mathbf{x}}_m$ $\hat{\mathbf{x}}_M$.

Шаг 5. Затем возвращаемся в старые координаты по формулам

$\mathbf{x}_m = S\hat{\mathbf{x}}_m,\ \mathbf{x}_M = S\hat{\mathbf{x}}_M$.

□

Вычислительная сложность решения складывается из вычислительной сложности решения проблемы собственных чисел матрицы размерности n, вычисления корня вещественной функции и решения n задач на экстремум.

Рассмотрим теперь специальный случай, когда полупространство вырождается в гиперплоскость (исходный предикат $P : \Sigma_{i=1}^{n} h_i x_i = c,\ h_i, c \in R$).

В этом случае может быть найдено более простое решение, так как достаточно найти первую точку пересечения решения задачи Коши (16) и гиперплоскости и проверить, не будет ли движение траектории скользящим по гиперплоскости.

Перейдем к новому базису и овеществим покомпонентные решения задачи Коши. Будем иметь n вещественных функций одной переменной $\hat{x}_i(t)$.

Подставим далее эти функции в исходный предикат $\hat{\mathbf{h}}\hat{\mathbf{x}} = c$, найдем первый неотрицательный корень t_* этого уравнения и вычислим соответствующие значения $\mathbf{x}(t_*) = S\hat{\mathbf{x}}(t_*)$.

Тогда искомые значения будут равны $t_m = t_M = t_*, \mathbf{x}_m = \mathbf{x}_M = \mathbf{x}(t_*)$.

Если корня t_* не существует, то будем иметь $t_m = t_M = \emptyset, \mathbf{x}_m = \mathbf{x}_M = \emptyset$.

Если начиная с некоторого момента t_1 уравнение $\hat{\mathbf{h}}\hat{\mathbf{x}} = c$ будет являться тождеством, то необходимо вернуться к решению задачи поиска экстремальных значений.

Опишем функцию "extrems" стиле последовательного уточнения. Через $in(P)$ и $bound(P)$ обозначается внутренность и граница области P соответственно. В алгоритме используются коэффициенты h_i предиката P.

Функция "extrems" в методе покоординатных оценок.

```
  const Infty+, ∅; // обозначение бесконечности и пустого множества
  type Real= set of {real ∪ Infty+ };
  type интервал : record {
  tmin : Real;
  tmax : Real;
  };
  type комплексное число : record {
  vreal : Real;
  vimage : Real;
  };
  type собственное значение : record {
  value : комплексное число; // собственное значение
  mult : int; // кратность собственного значения
  };
  type функция одной переменной: string; // явное представление функции
  type предикат : string;
  var n : int; // размерность автомата

func extrems (A: matrix[n,n] of real; b: vector[n] of Real;
P : предикат; x0 : vector[n] of Real) : {X1, X2: vector[n] of
Real, [T1,T2] : интервал }; // P ≡ ∑_{i=1}^{n} h_i x_i ~ c, ~= {<, ≤, =, ≥, >}
  var h̃ : vector[n] of Real; // вектор коэффициентов предиката
  var X1, X2, X̃1, X̃2 : vector[n] of Real; // локальные переменные
  var T, T1, T2 : Real; // локальные переменные
  var Λ : vector[n] of собственное значение;
  var G : matrix[n,n] of комплексное число;
  var P1 : предикат; // локальная переменная
```

```
  var x̃ : vector[n] of функция одной переменной; // вектор реше-
ний задачи Коши
  var flag : boolean; // флаг существования корня
  var k : целое;
        T1 := ∅; T2 := ∅;
        <решить проблему собственных значений матрицы A>;
        <описать решение задачи Коши в новом базисе x>;
        <описать область P̃ в новом базисе>;
        <составить предикат P̃(x̃)>;
        <найти первый интервал I=[T1,T2] истинности предиката
P̃(x̃)>;
        if T1=T2=0 then
              X̃1 := x̃0; X̃2 := x̃0;
        elseif P=∑_{i=1}^{n} h_i x_i = c and T1=T2 then
              X̃1 := x̃(T1); X2 := x̃(T1);
        else
              for k := 1 ... n
                    <исследовать функцию x̃[k] на экстремумы
в области t∈[T1,T2], результат записать в X̃1[k], X̃2[k]>;
              endfor;
        endif;
        <вернуться в старые координаты>;
  return (X1, X2, [T1,T2]).
<решить проблему собственных значений матрицы A>:
  {Λ, G } := eigenvalue(A).
<описать решение задачи Коши в новом базисе>:
  for k := 1 ... n
        x̃0[k] := ∑_{i=1}^{n} G[k,i]^{-1} x0[i];
        b̃[k] := ∑_{i=1}^{n} G[k,i]^{-1} b[i];
        x̃[k] := formula(Λ[k], b̃, x̃0[k]);
  endfor.
<описать область P в новом базисе>:
  for k := 1 ... n
        h̃[k] := ∑_{i=1}^{n} h_i G[i,k];
  endfor;
<составить предикат P̃(x̃(t))>:
  P̃ := "∑_{i=1}^{n} h̃[i] x̃[i] ∼ c".
```

```
<найти первый интервал I истинности предиката P̃(x̃)>:
  [T1,T2] := solve inequation(P̃(x̃)).
<исследовать функцию x̃[k](t) на экстремумы в области
t∈[T1,T2]>:
  {X̃1, X̃2} := solve extr(x̃[k], [T1,T2]).
<вернуться в старые координаты>:
  for k := 1 ...n
        X1[k] := ∑_{i=1}^{n} G[k,i] X̃1[i];
        X2[k] := ∑_{i=1}^{n} G[k,i] X̃2[i];
  endfor.

func solve inequation (Nq : неравенство) : интервал;
```

//Аргумент – неравенство, составленное подстановкой в исходный предикат P покомпонентных решений задачи Коши в новом базисе. Возвращает первую найденную область истинности неравенства (интервал времени).

```
  var x_0 : vector[n] of Real; // локальные переменные вектора
        if 0 ∈ in(Nq) then
        <найти первое значение T≥0, удовлетворяющее ¬Nq>;
              if <T не существует> then
                    T1 := 0; T2 := Infty+;
                    else T1 := 0; T2 := T;
              endif;
        endif;
        if 0 ∉ in(Nq) then
        <найти первое значение T≥0, удовлетворяющее Nq>;
              if <T не существует> then
                    T1 := T2 := ∅;
              else
                    <найти первое значение T2>T, удовлетво-
ряющее ¬Nq>;
                    if <T2 не существует> then
                    T1 := T; T2 := Infty+;
                    else T1 := T;
              endif;
        endif;
        if Nq=∑_{i=1}^{n} h_i x_i(t) ≤ (≥)c and 0 ∈ bound(Nq) then
        <найти первое значение T≥0, удовлетворяющее ¬Nq>;
```

```
            if <T не существует> then
                  T1 := 0;   T2 := Infty+;
            elseif T=0 then
                  return ([0,0]);
            else T1 := 0; T2 := T;
            endif;
      endif;
  return ([T1,T2]).
<найти первое значение T, удовлетворяющее неравенству P1>:
  { T , flag } := first time(P1).
<T не существует>:
  flag = false.
```

`func eigenvalue ( A : matrix[n,n] of real) : { L : set of собственное значение, G : matrix[n,n] of комплексное число };` //возвращает собственные значения заданной матрицы и матрицу преобразования к квазидиагональному виду, может быть реализована любым известным численным методом.

`func formula (`λ`: собственное значение, b: vector[n] of real,` x_0 `: real) : x : функция одной переменной;` //выбирает одну из формул овеществления (шаг 3 алгоритма) для компоненты решения задачи Коши в новом базисе в соответствие с заданным собственным значением λ

`func first time (Nq1 : неравенство) : {flag : boolean, : t: real};` //решает неравенство Nq1, возвращает первое вычисленное значение, если неравенство не имеет решение, то устанавливает flag = false, может быть реализована любым известным методом.

`func solve extr (x(t) : функция одной переменной, [T1,T2] : интервал ) : {X1, X2 : vector[n] of real};` // возвращает минимальное и максимальное значение вещественной функции на заданном интервале, может быть реализована любым известным методом.

2.4.4 Теорема о точности метода покоординатных оценок

Докажем теорему о прямоугольной области и интервале времени, построенных с помощью функции "Cauchy" обобщенного таймер-преобразования.

Теорема 2.4.2 *Прямоугольный параллелепипед* $D = [\mathbf{x}_{min}, \mathbf{x}_{max}]$, *построенный методом покоординатных оценок, является верхней оценкой области первого пересечения пучка решений задачи Коши (11)-(12) и полупространства (13), оптимальной в классе прямоугольных параллелепипедов.*

Интервал времени $\mathcal{I} = [t_{min}, t_{max}]$, *построенный методом покоординатных оценок, описывает точные временные границы, когда пучок решений задачи Коши (11)-(12) в первый раз будет находиться в полупространстве (13).*

Доказательство: Докажем вначале, что вся искомая область окажется внутри построенного параллелепипеда D.

Множество достижимости линейной системы ОДУ с выпуклым множеством начальных значений всегда остается выпуклым [21]. Более того, из теоремы 2.4.1 следует, что множество достижимости линейной системы ОДУ с постоянными коэффициентами и с прямоугольной начальной областью в любой момент времени представляет собой параллелепипед, вершинами которого являются точки, принадлежащие траекториям решений задач Коши, построенных из вершин начальной области.

При решении каждой точечной задачи Коши (функция "exstrems") используется линейное невырожденное преобразование координат, как известно, класс параллелепипедов инвариантен относительно афинных преобразований. Следовательно, при переходе к новому базису и обратно, множество достижимости не искривляется, при этом функция "solve extr" метода копоординатных оценок возвращает вычисленные экстремальные значения для каждого точечного решения задачи Коши, начинающегося в вершине прямоугольного параллелепипеда (12), в заданном полупространстве (13). Наконец, аппроксимируя полученную область прямоугольным параллелепипедом D по формулам (15), мы гарантируем, что все граничные точки (а значит и все другие) попадут в D.

Теперь покажем, что полученная аппроксимация является наилучшей возможной в классе прямоугольных параллелепипедов.

Предположим, что используемые численные методы позволяют вычислить точные экстремальные значения для каждого точечного решения задачи Коши, начинающегося в вершине прямоугольного па-

раллелепипеда (12), в заданном полупространстве (13). Это значит, что среди точечных решений задачи Коши (11) с начальными значениями $\mathbf{x}_0 \in D_0$ обязательно найдется хотя бы одно такое решение $\mathbf{x}^*(t)$, у которого некоторая компонента $x_i^*(t)$ в некоторый момент времени $t^* \in I$ коснется гиперплоскости $x_i - x_i^{(m)} = 0$ (являющейся нижней гранью D по координате x_i), а также найдется хотя бы одно решение $\mathbf{x}^{**}(t)$, у которого некоторая компонента $x_i^{**}(t)$ в некоторый момент времени $t^{**} \in I$ коснется гиперплоскости $x_i - x_i^{(M)} = 0$ (являющейся верхней гранью D по координате x_i) – для предикатов (13), где $\sim= \{\leq, =, \geq\}$. Аналогично, для предикатов (13), где $\sim= \{<, >\}$ среди точечных решений задачи Коши (11) с начальными значениями $\mathbf{x}_0 \in D_0$ найдется хотя бы одно решение $\mathbf{x}^*(t)$, у которого некоторая компонента $x_i^*(t)$ будет обладать свойством, что в некоторый момент времени $t^* \in I$ будет $sup\{x_i^*(t^*)\} \in \{x_i - x_i^{(m)} = 0\}$ и найдется хотя бы одно решение $\mathbf{x}^{**}(t)$, у которого некоторая компонента $x_i^{**}(t)$ будет обладать свойством, что в некоторый момент времени $t^* \in I$ будет $inf\{x_i^{**}(t^*)\} \in \{x_i - x_i^{(m)} = 0\}$. Это доказывает, что искомая область вписана в построенный прямоугольный параллелепипед, а значит получена оптимальная верхняя оценка искомой области в классе прямоугольных параллелепипедов.

Аналогично, если предположить, что используемые численные методы позволяют вычислить точные значения времени в функции "first time", то нетрудно показать, что интервал I, построенный по формулам 15, представляет собой точный временной интервал, когда пучок решений задачи Коши (11)-(12) в первый раз будет находиться в полупространстве (13). □

2.4.5 Замечание о методе покоординатных оценок

Метод покоординатных оценок ориентирован на построение верхней аппроксимации множества возможных фазовых состояний системы, соответствующих его пересечению с областью, ограниченной линейным предикатом. Это ограничивает возможность верификации вычислительной модели, полученной с помощью обобщенного таймер–преобразования. Для полной ее верификации необходимо построение как верхних, так и нижних аппроксимаций множеств достижимости в задачах оценивания.

Для построения нижних аппроксимаций достаточно модифицировать метод покоординатных оценок, например, следующим образом:

после нахождения экстремальных значений $\mathbf{x}_{min}^{(1)}, \mathbf{x}_{max}^{(1)}, \ldots, \mathbf{x}_{min}^{(m)}, \mathbf{x}_{max}^{(m)}$ граничных решений задачи Коши (11)-(12) и соответствующих временных интервалов $t_{min}^{(1)}, t_{max}^{(1)}, \ldots, t_{min}^{(m)}, t_{max}^{(m)}$ ($m = 2^n$, функция "exstrems"), неоходимо выбрать в качестве $\mathbf{x}_{max}$ минимальное значение из максимальных, а в качестве $\mathbf{x}_{min}$ – максимальное значение из минимальных:

$$\mathbf{x}_{min} = max\{\mathbf{x}_{min}^{(1)}, \ldots, \mathbf{x}_{min}^{(m)}\}, \quad \mathbf{x}_{max} = min\{\mathbf{x}_{max}^{(1)}, \ldots, \mathbf{x}_{max}^{(m)}\}$$
$$t_{min} = max\{t_{min}^{(1)}, \ldots, t_{min}^{(m)}\}, \quad t_{max} = min\{t_{max}^{(1)}, \ldots, t_{max}^{(m)}\}$$

В этом случае построенная прямоугольная область будет целиком содержаться внутри искомой области, то есть будет являться ее нижней аппроксимацией. Возможны также другие решения.

С помощью обобщенного таймер–преобразования с использованием модифицированного метода копоординатных оценок будет построен таймированный автомат T_H^-, который будет обладать свойством:

$$Reach(T_H^-) \subseteq Reach(H)$$

Автомат H будет допускать все вычисления, допустимые в автомате T_H^-, построенным из H с помощью обобщенного таймер–преобразования с модифицированным методом покоординатных оценок, однако часть траекторий, допустимых в исходном автомате H, не допустима в автомате T_H^-. Следовательно, алгоритм обобщенного таймер–преобразования с модифицированным методом покоординатных оценок позволяет верифицировать те требования к поведению, результат которых отрицателен.

Построение двухсторонних аппроксимаций в методе покоординатных оценок сделает возможным позволит проводить полную верификацию качественных свойств исходного гибридного автомата H на его двух линейных аппроксимациях T_H^- и T_H^+.

2.5 Метод оценки временного интервала. Функция "Cauchy1"

2.5.1 Постановка задачи и описание метода

В данном разделе предложено решение частной задачи нахождения временного интервала, соответствующее пересечению пучка решений

задачи Коши (11-12) и области фазового пространства системы, ограниченной предикатом P.

Сформулируем эту задачу.

Пусть дана линейная система дифференциальных уравнений с постоянными коэффициентами размерности n:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + \mathbf{b}; \quad a_{ij}, b_i \in R \tag{21}$$

Область начальных значений задана прямоугольным параллелепипедом в R^n:

$$\mathbf{x}_0 \in D_0 = [\ L, U\ ]; \quad [l_i, u_i] \in \mathcal{R}, \tag{22}$$

Требуется оценить нижнюю и верхнюю временные границы t_1, t_2, когда пучок решений задачи Коши (11-12) $\mathbf{x}(t)$ в первый раз будет удовлетворять предикату

$$P \ : \quad \Sigma_{i=1}^{n} h_i x_i \leq c, \ h_i, c \in R \tag{23}$$

(будет находиться в области фазового пространства системы, ограниченной неравенством $\Sigma_{i=1}^{n} h_i x_i \leq c$).
Перечислим основные шаги решения данной задачи.

Шаг 1. (Вычисление собственных значений матрицы A и элементов матрицы S преобразования координат.) Предположим, что мы нашли невырожденное преобразование координат S, приводящее матрицу A к квазидиагональному виду Λ. Если система дифференциальных уравнений (21) однородна, то получим решение задачи Коши в новых координатах:

$$\hat{\mathbf{x}} = e^{\Lambda t}\hat{\mathbf{x}}_0 \tag{24}$$

где $\hat{\mathbf{x}}$ – вектор переменных системы ОДУ в новом базисе ($\hat{\mathbf{x}} = S^{-1}\mathbf{x}$), $\hat{\mathbf{x}}_0$ – начальная точка в новом базисе ($\hat{\mathbf{x}}_0 = S^{-1}\mathbf{x}_0$), Λ – матрица, которая в общем случае составлена из s жордановых ящиков, соответствующих различным собственным значениям λ_i, $i = 1 \ldots s$ кратности m_i.

Для неоднородной системы ОДУ будем иметь (если существует A^{-1}):

$$\hat{\mathbf{x}}(t) = e^{\Lambda t}\hat{\mathbf{x}}_0 + A^{-1}(e^{\Lambda t} - I)\hat{\mathbf{b}}, \tag{25}$$

где $\hat{\mathbf{b}} = S^{-1}\mathbf{b}$.

Если среди собственных значений λ_i матрицы A найдется хотя бы одна комплексно–сопряженная пара $\lambda_{i,i+1}$, то функции $\hat{x}_i(t)$ будут комплексно-значные ($s_{ij},\ \hat{b}_i,\ \hat{x}_{i0}\ \in C$).

Шаг 2. (Преобразование области начальных значений и области, описанной предикатом, при переходе к ортогональному базису.) Поскольку преобразование S линейное, то область начальных значений в новых координатах будет представлять собой параллелепипед в R^n (класс параллелепипедов инвариантен относительно афинных преобразований), который мы аппроксимируем прямоугольным параллелепипедом, преобразуя компоненты вектора начальных значений по правилам интервального анализа [2]:

$$\hat{x}_{i0} = \sum_{j=1}^{n} s_{ij}^{-1} x_{j0}, \quad x_{j0} \in [l_j;\ u_j], \quad i = 1 \ldots n$$

или в векторном виде:

$$\hat{\mathbf{x}}_0 \in [\ \hat{L}\ ,\ \hat{U}\ ] \tag{26}$$

Заметим, что таким образом мы получим оптимальную верхнюю аппроксимацию области начальных значений в новом базисе в классе прямоугольных параллелепипедов.

Предикат P определяет полупространство, ограниченное гиперплоскостью, которое при замене переменных x_i на $\sum_{j=1}^{n} s_{ij}\hat{x}_j$, в новом базисе будет описываться следующим очевидным образом:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_i s_{ij} \hat{x}_j \le c$$

или в векторном виде:

$$\hat{P} : \hat{\mathbf{h}}\hat{\mathbf{x}}\ \le\ c, \quad \hat{\mathbf{h}} = \mathbf{h}^T S \tag{27}$$

где s_{ij} – элементы матрицы преобразования S, $\quad \hat{h}_j = \sum_{i=1}^{n} h_i s_{ij}$.

Шаг 3. (Описание предиката в новом базисе интервально-значной вещественной функцией одной переменной). Овеществим решение задачи Коши (см. шаг 3 метода покоординатных оценок) и составим функцию $f(t)$, подставив покомпонентные решения задачи Коши в левую часть неравенства (27). Функция $f(t)$ является вещественной функцией переменной t с интервально-значными константами $\hat{\mathbf{x}}_0 \in [\ \hat{L}\ ,\ \hat{U}\ ]$.

Вид функции f однозначно определяется собственными значениями матрицы A, для однородной системы ОДУ, она будет иметь вид:

1) в случае попарно-различных вещественных собственных значений

$$f(t) = \sum_{i=1}^{n} \tilde{x}_{io} e^{\lambda_i t} \tag{28}$$

2) для жордановых ящиков размером m_k, соответствующих вещественным собственным значениям λ_k кратности $m_k > 1$ (всего s различных собственных значений),

$$f(t) = \sum_{i=1}^{s} P_{m_k-1}(t) e^{\lambda_i t} = \sum_{k=1}^{s} \sum_{j=0}^{m_k-1} \tilde{x}_{(k+j)0} t^j e^{\lambda_k t} \tag{29}$$

3) при существовании среди попарно-различных собственных значений комплексно-сопряженных пар $\lambda_{i,i+1} = \alpha^{(i)} \pm i\beta^{(i)}$ соответствующие слагаемые в формуле (28) заменяются слагаемыми вида

$$\tilde{x}_{i0}\ e^{\alpha^{(i)} t} cos(\beta^{(i)} t) + \tilde{x}_{(i+1)0}\ e^{\alpha^{(i)} t} sin(\beta^{(i)} t) \tag{30}$$

4) при существовании среди собственных значений комплексно-сопряженных пар $\lambda_{i,i+1} = \alpha \pm i\beta$ кратности $m_k > 1$ соответствующие слагаемые в формуле (28) заменяются квазимногочленами вида

$$\sum_{j=0}^{m_k-1} \tilde{x}_{j0}^{(i)} t^j e^{\alpha t} cos(\beta t)\ +\ \sum_{l=0}^{m_k-1} \tilde{x}_{l0}^{(i)} t^l e^{\alpha t} sin(\beta t) \tag{31}$$

Во всех случаях, очевидно, $\tilde{x}_{io} = [a_i, c_i]$ – интервально-значные константы, которые однозначно определяются начальными условиями (с учетом правила $(-1)[a, b] = [-b, -a]$).

Для неоднородной системы ОДУ функция $f(t)$ может быть составлена аналогичным образом (в соответствии с формулами, полученными в шаге 3 метода покоординатных оценок).

Шаг 4. (Нахождение интервала времени первого пересечения интервально-значной вещественной функции f границы полуплоскости, описанной предикатом в новом базисе). Построим для интервально–значной вещественной функции $f(t)$ граничные вещественные функциями f_1, f_2. Согласно правилам, разработанным в интервальном анализе (см. [22]), соответствующие формулы граничных функций

будут иметь следующий вид (ограничимся рассмотрением однородной системы ОДУ):

1) для случая попарно-различных собственных значений (формула (28))

$$f_1(t) = \sum_{i=1}^{n} a_i e^{\lambda_i t}$$

$$f_2(t) = \sum_{i=1}^{n} c_i e^{\lambda_i t}$$

2) для жордановых ящиков размером m_k, соответствующих вещественным собственным значениям λ_k, в области $t \geq 0$ (всего s различных собственных значений, формула (29)),

$$f_1(t) = \sum_{k=1}^{s} \sum_{j=0}^{m_k-1} a_{(k+j)0} t^j e^{\lambda_k t}$$

$$f_1(t) = \sum_{k=1}^{s} \sum_{j=0}^{m_k-1} c_{(k+j)0} t^j e^{\lambda_k t}$$

3) при существовании среди собственных значений комплексно-сопряженных пар $\lambda_{i,i+1} = \alpha^{(i)} \pm i\beta^{(i)}$ (формула (30))

$$f_1(t) = \sum_{i\in 1\ldots n}^{\lambda_i \in R} a_i e^{\lambda_i t} + \sum_{i\in 1\ldots n}^{\lambda_{i,i+1}\in C} e^{\alpha^{(i)}t}[\ (a_i\theta_{cos}(i) + c_i\gamma_{cos}(i))cos(\beta^{(i)}t) +$$

$$+(a_{i+1}\theta_{sin}(i) + c_{i+1}\gamma_{sin}(i))sin(\beta^{(i)}t)\ ],$$

$$f_2(t) = \sum_{i\in 1\ldots n}^{\lambda_i \in R} c_i e^{\lambda_i t} + \sum_{i\in 1\ldots n}^{\lambda_{i,i+1}\in C} e^{\alpha^{(i)}t}[\ (a_i\gamma_{cos}(i) + c_i\theta_{cos}(i))cos(\beta^{(i)}t) +$$

$$+(a_{i+1}\gamma_{sin}(i) + c_{i+1}\theta_{sin}(i))sin(\beta^{(i)}t)\ ]$$

4) при существовании среди собственных значений комплексно-сопряженных пар $\lambda_{i,i+1} = \alpha^{(i)} \pm i\beta^{(i)}$ кратности m_i, в области $t \geq 0$, всего s различных собственных значений (формула (31)),

$$f_1(t) = \sum_{i\in 1\ldots n}^{\lambda_i \in R} a_i e^{\lambda_i t} + \sum_{i\in 1\ldots n}^{\lambda_{i,i+1}\in C} e^{\alpha^{(i)}t}[\ \sum_{j=0}^{m_k-1} t^j(a_j^{(i)}\theta_{cos}(i) + c_j^{(i)}\gamma_{cos}(i))cos(\beta^{(i)}t) +$$

$$+ \sum_{l=0}^{m_k-1} t^l(a_l^{(i)}\theta_{sin}(i) + c_l^{(i)}\gamma_{sin}(i))sin(\beta^{(i)}t)\ ],$$

$$f_2(t) = \sum_{i\in 1...n}^{\lambda_i\in R} c_i e^{\lambda_i t} + \sum_{i\in 1...n}^{\lambda_{i,i+1}\in C} e^{\alpha^{(i)}t}[\sum_{j=0}^{m_k-1} t^j(a_j^{(i)}\gamma_{cos}(i) + c_j^{(i)}\theta_{cos}(i))cos(\beta^{(i)}t) +$$

$$+ \sum_{l=0}^{m_k-1} t^l(a_l^{(i)}\gamma_{sin}(i) + c_l^{(i)}\theta_{sin}(i))sin(\beta^{(i)}t)\ ].$$

Функции $\theta_{cos}(i), \gamma_{cos}(i), \theta_{sin}(i), \gamma_{sin}(i)$ определяют выбор констант a_i, c_i в зависимости от знаков функций $cos(\beta t)$ и $sin(\beta t)$. Они определяются по следующим формулам:

$$\theta_{cos}(i) = \frac{1 + sgn(cos\beta^{(i)}t)}{2} = \begin{cases} 1, & \text{если } cos\beta^{(i)}t > 0 \\ 0, & \text{если } cos\beta^{(i)}t < 0 \end{cases}$$

$$\gamma_{cos}(i) = 1 - \theta_{cos}(i) = \begin{cases} 0, & \text{если } cos\beta^{(i)}t > 0 \\ 1, & \text{если } cos\beta^{(i)}t < 0 \end{cases}$$

$$\theta_{sin}(i) = \frac{1 + sgn(sin\beta^{(i)}t)}{2} = \begin{cases} 1, & \text{если } sin\beta^{(i)}t > 0 \\ 0, & \text{если } sin\beta^{(i)}t < 0 \end{cases}$$

$$\gamma_{sin}(i) = 1 - \theta_{sin}(i) = \begin{cases} 0, & \text{если } sin\beta^{(i)}t > 0 \\ 1, & \text{если } sin\beta^{(i)}t < 0 \end{cases}$$

Для вычисления всех возможных значений t, при которых интервально–значная функция $f(t)$ попадет в нижнюю полуплоскость, ограниченную прямой $t = c$ (исходный предикат $\sum_{i=1}^{n} h_i x_i < (\leq) c$), необходимо исследование граничной функции $f_1(t)$, в то время как для вычисления значений t, при которых $f(t)$ попадет в верхнюю полуплоскость (исходный предикат $\sum_{i=1}^{n} h_i x_i > (\geq) c$), необходимо исследование граничной функции $f_2(t)$. Рассмотрим возможные варианты для нижней полуплоскости:

1) Если у уравнения $f_1(t) = c, R_{\geq 0}$ нет вещественных корней и $f_1(0) < c$, то неравенство выполняется всегда (предикат P на соответствующей дуге автомата заменяется на $True$);

Если у уравнения $f_1(t) = c, R_{\geq 0}$ нет вещественных корней и $f_1(0) > c$, то неравенство никогда не будет выполнено (предикат P на соответствующей дуге автомата заменяется на $False$);

Если у уравнения $f_1(t) = c, R_{\geq 0}$ нет вещественных корней и $f_1(0) = c$, то необходимо более подробное исследование $f_1(t)$ вблизи точки $t =$

0 (с помощью производных) и выбор одного из предыдущих случаев в зависимости от знака функции (производных) в окрестности нуля.

2) Пусть r_0, r_1 – два первых положительных вещественных корня уравнения $f_1(t) = c$. Тогда

если $f_1(0) < c$, то неравенство в первый раз выполняется при $t \in [0; r_0]$;

если $f_1(0) > c$, то неравенство в первый раз выполняется при $t \in [r_0; r_1]$;

если $f_1(0) = c$, то необходимо более подробное исследование $f_1(t)$ вблизи точки $t = 0$ (исследование производных) и выбор одного из предыдущих случаев в зависимости от знака функции (производных) в окрестности нуля.

Подобным образом могут быть получены временные границы для предиката $\sum_{i=1}^{n} h_i x_i \geq c$ с вычислением двух первых положительных вещественных корней уравнения $f_2(t) = c$.

Специальный случай представляет собой предикат $\sum_{i=1}^{n} h_i x_i = c$ (полуплоскость вырождается в прямую). Для этого случая необходимо найти первые неотрицательные вещественные корни для обоих уравнений $f_1(t) = c, f_2(t) = c$. □

Вычислительная сложность решения частной задачи Коши складывается, таким образом, из вычислительной сложности решения проблемы собственных чисел матрицы размерности n и отыскания двух первых корней вещественной функции.

Опишем функцию "Cauchy1" в стиле последовательного уточнения. Последним аргументом функции является дуга $e \in E$, для которой производятся вычисления, или 0 – если определяется длительность нахождения в локации по предикату-инварианту.

Функция "Cauchy1" в алгоритме обобщенного таймер-преобразования.

```
const Infty+, ∅; // обозначение бесконечности и пустого множества
type Real= set of {real ∪ Infty+ };
type прямоугольная область : record {
L : vector[n] of Real;
U : vecrot[n] of Real;
};
type интервал : record {
```

```
  t_min : Real;
  t_max : Real;
  };
  type комплексное число : record {
  vreal : Real;
  vimage : Real;
  };
  type собственное значение : record {
  value : комплексное число; // собственное значение
  mult : int; // кратность собственного значения
  };
  type функция одной переменной: string; // явное представление функции
  type предикат: string; // линейный предикат
  type неравенство : string; // линейное неравенство
  var n : int; // размерность автомата

func Cauchy1 (A : matrix[n,n] of real; b: vector[n] of real;
e: дуга) : {[T1,T2]: интервал}; // P ≡ ∑_{i=1}^{n} h_i x_i ~ c, ~= {<, ≤, =, ≥, >}
  var P : предикат;
  var h̃ : vector[n] of Real; // вектор коэффициентов предиката в новом базисе
  var T, T1, T2, T1', T2' : Real; // локальные переменные
  var Λ : vector[n] of собственное значение; // массив собственных значений
  var G : matrix[n,n] of комплексное число; // матрица преобразования
  var x̃ : vector[n] of функция одной переменной; // вектор решений задачи Коши в новом базисе
  var X̃ : прямоугольная область; // область начальных значений в новом базисе
  var flag : boolean; // флаг существования корня
  var k : целое;
        if e≠ 0 then P := e.pre else P :=v.inv;
        endif;
        T1 := ∅; T2 := ∅;
```

```
        <решить проблему собственных значений матрицы A>;
        <описать решение задачи Коши в новом базисе>;
        <описать область P в новом базисе - P̃>;
        <составить интервально-значную функцию f=left[P̃]>
        <составить граничные функции f - f1 и f2>;
        case P=∑_{i=1}^n h_i x_i < (≤) c :
              if f1(0) < (≤) c then // нач.точка внутри области P
              <найти первое значение T≥0, удовлетворяющее
неравенству "f1(t) ≥ (>) c">;
              if <T не существует> then
                    T1 := 0; T2 := Infty+;
                    else T1 := 0; T2 := T;
              endif;
              return [T1, T2]; // [T1, T2) для ' <'
        endif;
        if f1(0) ≥ (>) c then // нач.точка вне области P
              if e = 0 then return("never");
  // нач.точка не удовлетворяет инварианту
              endif;
              <найти первое значение T≥0, удовлетворяющее
неравенству "f1(t) < (≤) c">;
              if <T не существует> then
                    T1 := T2 := ∅;
              else
              <найти первое значение T2>T, удовлетворяющее
неравенству "f1(t) ≥ (>) c">;
                    if <T2 не существует> then
                    T1 := T; T2 := Infty+;
                    else T1 := T;
              endif;
              return [T1, T2]; // (T1, T2) для ' <'
        endif;
        case P=∑_{i=1}^n h_i x_i > (≥) c :
              if f2(0) > (≥) c then // нач.точка внутри области P
              <найти первое значение T≥0, удовлетворяющее
неравенству "f2(t) ≤ (<) c">;
              if <T не существует> then
```

```
                    T1 := 0; T2 := Infty+;
                    else T1 := 0; T2 := T;
               endif;
               return [T1, T2]; // [T1, T2) для ' >'
          endif;
          if f2(0) ≤ (<) c then // нач.точка вне области P
               if e = 0 then return("never");
  // нач.точка не удовлетворяет инварианту
               endif;
               <найти первое значение T≥0, удовлетворяющее
неравенству "f2(t) > (≥)c">;
               if <T не существует> then
                    T1 := T2 := ∅;
               else
               <найти первое значение T2>T, удовлетворяющее
неравенству "f2(t) ≤ (<)c">;
                    if <T2 не существует> then
                    T1 := T; T2 := Infty+;
                    else T1 := T;
               endif;
               return [T1, T2]; // (T1, T2) для ' >'
          endif;
          case P=∑_{i=1}^{n} h_i x_i = c :
               <найти первый интервал [T1,T2] истинности
неравенства f1(t)= c)>;
               <найти первый интервал [T1',T2'] истинности
неравенства f2(t)= c)>;
               T1 := min {T1, T1'}; T2 := max {T2, T2'};
               if T1 ≠ 0 and e = 0 then return("never");
  // нач.точка не удовлетворяет инварианту
          endcase;
          return ([T1,T2]).
<решить проблему собственных значений матрицы A>:
  {Λ, G } := eigenvalue(A).
<описать решение задачи Коши в новом базисе>:
  for k := 1 ... n
          X̃.L[k] := ∑_{i=1}^{n} G[k,i]^{-1} e.X_e.L[i];
```

```
        X̃.U[k] := Σ_{i=1}^{n} G[k,i]^{-1} e.X_e.U[i];
        b̃[k] := Σ_{i=1}^{n} G[k,i]^{-1} b[i];
        x̃[k] := formula(Λ[k], b̃, X̃);
  endfor.
<описать область P в новом базисе>:
  for k := 1 ... n
        h̃[k] := Σ_{i=1}^{n} h_i G[i,k];
  endfor;
<составить интервально-значную функцию f=left[P]>:
  f := Σ_{i=1}^{n} h̃[i] x̃[i].
<составить граничные функции f - f1 и f2>:
  f1 := formulaf1( f, X̃);
  f2 := formulaf2( f, X̃).
<найти первое значение T, удовлетворяющее неравенству P1>:
  { T , flag } := first time(P1).
<T не существует>:
  flag = false.
<найти первый интервал [T1,T2] истинности неравенства Nq)>:
  [T1,T2] := solve inequation ( Nq ).
```

---

func eigenvalue (A : matrix[n,n] of real) : { L : set of собственное значение, G : matrix[n,n] of комплексное число };

//возвращает собственные значения заданной матрицы и матрицу преобразования к квазидиагональному виду, может быть реализована любым известным численным методом.

func formula (λ : собственное значение, $\mathbf{b}$: vector[n] of real; x_0 : real) : x : функция одной переменной; //выбирает одну из формул овеществления (??) для компоненты решения задачи Коши в новом базисе в соответствие с заданным собственным значением λ

func formulaf1 (f: функция одной переменной, X: прямоугольная область) : функция одной переменной; //выбирает одну из формул (шаг 4 алгоритма) для нижней граничной функции интервально-значной функции f

func formulaf2 (f: функция одной переменной, X: прямоугольная область) : функция одной переменной; //выбирает одну из формул (шаг 4 алгоритма) для верхней граничной функции интервально-значной функции f

`func first time (Nq1 : неравенство) : {flag : boolean, t : real};` //решает неравенство Nq1, возвращает первое вычисленное значение, если неравенство не имеет решение, то устанавливает flag = false, может быть реализована любым известным методом.

`func solve inequation( Nq : неравенство) : интервал;` //Аргумент – неравенство, составленное подстановкой в исходный предикат P покомпонентных решений задачи Коши в новом базисе. Возвращает первую найденную область истинности неравенства (интервал времени), функция описана в п.2.4

2.5.2 Теорема о точности метода оценки временного интервала

Докажем теорему об интервале времени, построенном с помощью функции "Cauchy1" обобщенного таймер-преобразования.

Теорема 2.5.1 *Интервал времени $\mathcal{I} = [t_1, t_2]$, построенный методом оценки временного интервала, содержит в себе интервал, когда пучок решений задачи Коши (11)-(12) в первый раз будет находиться в полупространстве (13).*

Доказательство: Из теоремы 2.4.1 следует, что множество достижимости линейной системы ОДУ с постоянными коэффициентами и с прямоугольной начальной областью в любой момент времени представляет собой параллелепипед, вершинами которого являются точки, принадлежащие траекториям решений задач Коши, построенных из вершин начальной области. При переходе к новому базису используется линейное невырожденное преобразование координат. Класс параллелепипедов инвариантен относительно афинных преобразований. Следовательно, прямоугольная область начальных значений в новом базисе будет являться косоугольным параллелепипедом. Используя в методе оценки временного интервала правила интервального анализа для преобразования области начальных значений (формула (26)), мы расширяем косоугольный параллелепипед до прямоугольного параллелепипеда.

Согласно правилам интервального анализа [22], построенные граничные функции $f_1(t), f_2(t)$ описывают экстремальные поведения интервально-значной функции f. Предположим, что используемые численные методы позволяют вычислить точные значения времени

с помощью функции "first time" метода оценки временного интервала для граничных функций. Это значит, что в новом базисе будут найдены точные временные границы, когда пучок решений задачи Коши с расширенной областью начальных значений в первый раз будет находиться в полупространстве (23).

Следовательно, весь искомый интервал времени окажется внутри построенного интервала $\mathcal{I}$. □

Заключение

Работа посвящена исследованию непрерывно–дискретных систем и созданию новых методов моделирования и качественного анализа для этого класса сложных систем.

Основные результаты работы сводятся к следующему:

1. Для класса гибридных автоматов с линейными системами дифференциальных уравнений с постоянными коэффициентами и ограниченной интервальной инициализацией разработан алгоритм линейной аппроксимации "обобщенное таймер-преобразование". Границы разрешимости символьной верификации расширены до подкласса непрерывно-дискретных систем с линейными системами дифференциальных уравнений с постоянными коэффициентами и ограниченной интервальной инициализацией. Расширение класса достигнуто благодаря обогащению гибридного подхода методами численного и интервального анализа. Таким образом, подтверждена возможность и целесообразность объединения численных и дискретных подходов к анализу непрерывно-дискретных систем.

2. Сформулированы и теоретически обоснованы условия применимости разработанного алгоритма для класса гибридных автоматов с линейными системами дифференциальных уравнений и ограниченной интервальной инициализацией и для их параллельной композиции.

3. Доказаны теоремы о сходимости метода символьной верификации для линейной аппроксимации гибридного автомата с линейными системами дифференциальных уравнений, построенной с помощью обобщенного таймер-преобразования, и для линейной аппроксимации параллельной композиции гибридных автоматов с линейными системами дифференциальных уравнений.

4. Доказана теорема о корректности положительных результатов символьной верификации для гибридных автоматов с линейными системами дифференциальных уравнений и их параллельной композиции.

Создание алгоритма обобщенного таймер-преобразования является первой попыткой практической реализации идеи объединения методов непрерывного и дискретного моделирования в единую технологию моделирования и качественного анализа при исследовании непрерывно–дискретных систем. Идея вычисления длительности нахождения в локациях гибридного автомата (и оценки значений фазовых переменных) с помощью методов теории ОДУ и интервального анализа, заложенная в алгоритме обобщенного таймер–преобразования, доказывает возможность и целесообразность такого объединения.

Алгоритм обобщенного таймер-преобразования позволяет значительно расширить возможности различных существующих систем компьютерного моделирования.

В настоящее время начата реализация алгоритма обобщенного таймер-преобразования в системе компьютерного моделирования Model Vision 3.0. В процессе работы подтвердилось предположение о том, что метод символьной верификации совместно с алгоритмом обобщенного таймер-преобразования в численных компьютерных системах моделирования может значительно упростить процесс моделирования непрерывно-дискретных систем, так как в этом случае становится возможным проверка качественных свойств поведения системы и параметрический интервальный анализ на предварительном этапе моделирования, а также позволит исследовать вопросы параллелизма и взаимодействия компонент непрерывно-дискретной системы. В компьютерных системах дискретного моделирования, таких как Covers, реализация алгоритма позволит проводить качественный анализ поведения непрерывно–дискретных систем чисто дискретными методами, если выполнены условия приводимости анализируемой системы к системе временных переходов. В компьютерных системах моделирования, основанных на символьной верификации, таких как HyTech, реализация алгоритма значительно расширит класс непрерывно–дискретных систем, для которых возможна верификация.

В процессе разработки алгоритма предложен метод покоординатных оценок для задачи оценивания фазового состояния линейной системы ОДУ. Реализация этой части алгоритма в компьютерных системах моделирования позволит задавать начальные условия в виде интервалов при моделировании и анализе непрерывно–дискретных систем с линейными системами ОДУ.

На основании разработанного алгоритма прослеживается путь решения задачи линеаризации для более широкого класса гибридных автоматов (расширение алгоритма обобщенного таймер-преобразования для случая неограниченных интервалов). Последние работы по интервальному анализу [16] дают основание надеяться на положительный результат в этом направлении.

Усовершенствованием обобщенного таймер–преобразования может стать добавление в метод покоординатных оценок блока для построения нижних аппроксимаций множества возможных фазовых состояний линейной системы ОДУ. Построение двухсторонних аппроксимаций множеств достижимости линейных систем ОДУ позволит проводить полную верификацию качественных свойств гибридных автоматов описанного класса с помощью алгоритма обобщенного таймер–преобразования.

Список литературы

[1] Андронов А.А., Витт А.А., Хайкин С.Э.: Теория колебаний. М: Гос.изд.ф.-м.лит-ры, 1959, 915с.

[2] Алефельд Г., Херцбергер Ю.: Введение в интервальные вычисления. Москва, "Мир", 1987, 354с.

[3] Арнольд В.И.: Обыкновенные дифференциальные уравнения. Москва, "Наука", 1971, с.240.

[4] Баутин Н.Н.: Динамические модели часовых ходов. В сб. "Памяти А.А.Андронова", Изд. АН СССР, 1955.

[5] Баутин Н.Н.: Теория точечных преобразований и динамическая теория часов. Труды междун.симп. по нелин. калеб. Изд. АН УССР, Киев, 1963.

[6] Беллман Р.: Введение в теорию матриц. Москва, "Наука", 1969, с.368.

[7] Борщев А., Карпов Ю., Колесов Ю.: Спецификация и верификация систем логического управления реального времени. В сб. "Системная информатика", вып.2, ИСИ СО РАН, Н-ск, 1993, 40с.

[8] Бусленко Н.П.: Моделирование сложных систем. М: "Наука", 1978.

[9] Бутенин Н.В., Неймарк Ю.И., Фуфаев Н.А.: Введение в теорию нелинейных колебаний. М: "Наука", 1976, 384с.

[10] Емельянов В.С.: Введение в системное моделирование. М:"Мир",1985, с.

[11] Калман Р., Фалб П., Арбиб М.: Очерки по математической теории систем. М: "Мир", 1971, 700с.

[12] Карпов Ю.Г., Парийская Е.: Качественный анализ гибридных систем. В сб."Вычислительные, измерительные и управляющие системы". Труды СПбГТУ N 449, с. 16–20. С.-Петербург, 1994г.

[13] Колесов Ю.Б., Парийская Е.Ю.,Сениченков Ю.Б.: Построение, решение и анализ свойств систем алгебро–дифференциальных

уравнений в пакете MODEL VISION 3.0. Вторая международная н.-т. конференция "Дифференциальные уравнения и приложения". Тезисы докладов, с.122–124, С.-Петербург, 1998г.

[14] Корноушенко Е.К.: Интервальные покоординатные оценки для множеств достижимых состояний линейной стационарной системы. В ж. Автоматика и телемеханика, 1980, N 5 с.12–22, N 12 с.10–17.

[15] Неймарк Ю.И.: Метод точечных отображений в теории нелинейных колебаний. М: "Наука", 1972, 471с.

[16] Нестеров В.М.: Об одном обобщении интервального анализа и его применении для оценки множества значений функции. В сб. "Математические методы построения и анализа алгоритмов", с.109–124. Ленинград, "Наука", 1990г.

[17] Первозванский А.А.: Курс теории автоматического управления. М: "Наука", 1986, 615с.

[18] Прицкер А.: Введение в имитационное моделирование и язык СЛАМ II. М: "Мир", 1987, 646с.

[19] Программное обеспечение моделирования непрерывно-дискретных систем. (под ред. В.Глушкова), М: "Наука", 1975.

[20] Теория систем. Математические методы и моделирование. Сб. статей под ред. А.Колмогорова, С.Новикова. М: "Мир", 1989, 384с.

[21] Черноусько Ф.Л.: Оценивание фазового состояния динамических систем. Метод эллипсоидов. М.: Наука, 1988, 320с.

[22] Шокин Ю.И.: Интервальный анализ. Новосибирск, "Наука", 110с., 1981г.

[23] Парийская Е.: Сравнительный анализ математических моделей и подходов к моделированию и анализу непрерывно–дискретных систем. Эл.ж. рег.N П23275 "Дифференциальные уравнения и процессы управления", http://www.diff.Alpha.iiep.csa.ru, 1 ,1997, 25с.

[24] Парийская Е.: Применение методов теории реактивных систем в задачах моделирования и качественного анализа непрерывно–дискретных систем. Эл.ж.

рег.N П23275 "Дифференциальные уравнения и процессы управления", http://www.diff.Alpha.iiep.csa.ru, 2 ,1998, 42с.

[25] Парийская Е.: Гибридный подход к моделированию и качественному анализу динамических систем. Алгоритмы линейной аппроксимации нелинейного гибридного автомата. Труды $2^{-ой}$ межд. н.-т. конф. "Дифференциальные уравнения и приложения", с.174–177, С.-Петербург, 1998г.

[26] Парийская Е.: Основные идеи новой технологии моделирования и анализа сложных непрерывно–дискретных систем. Материалы н.-т. конф. "Фундаментальные исследования в технических университетах", с.169–170, С.-Петербург, 1997г.

[27] Парийская Е.: Применение гибридного автомата в задачах качественного анализа динамических систем. Материалы н.-т. конф. "Фундаментальные исследования в технических университетах", с.106–107, С.-Петербург, 1998г.

[28] Alur R., Fix L., Henzinger T.A.: A Determinizable Class of timed Automata. 6^{th} International Conference CAV'94. Lecture Notes in Comp. Sci. 818, p.1–13, 1994.

[29] Alur R., Courcoubetis C., Dill D.L.: Model-Checking for real-time systems. 5th LICS (5th IEEE Simp. Logic in Comp. Sci.), p.414-425. IEEE Comp. Soc. Press, 1990.

[30] Alur R., Courcoubetis C., Henzinger T., Ho P-T.: Hybrid automata: an algorithmic approach to the specification and analysis of hybrid systems. In Workshop on Theory of Hybrid Systems,Lyndby, Denmark, June 1993. LNCS 736, Springer-Verlag.

[31] Alur R., Courcoubetis C., Halbwachs N., Henzinger T.A., Ho P.-H., Olivero A., Sifakis J., Yovine S.: The algorithmic analysis of hybrid systems. Theoretical Comp.Sc.,138, p.3–34, 1995.

[32] Alur R, Feder T, Henzinger T.A.: The benefits of relaxing punctuality. In Proceeding of the Annual Symp. on Principles of Distributed Computing, p.139–152, ACM Press, 1991.

[33] Alur R., Henzinger T.A., Ho P.-H.: Automatic symbolic verification of embedded dydtems. IEEE Transaction on Software Engineering, 22(3), p.181–201, 1996.

[34] Asarin E., Maler O., Pnueli A.: Reachability Analysis of Dinamical Systems having Piecewise-Constant Derivatives. Theoretical Comp. Sci., v.138,N.1, 1995

[35] Asarin E., Maler O.: On some Relations between Dynamical Systems and Transition Systems. Proc. of ICALP'94, Lecture Notes in Comp. Sci 820, p.59–72, Springer-Verlag, 1994.

[36] Bertolomieu B., Diaz M.: Modeling and verification of time dependent systems using time Petri nets. IEEE Transactions on Software Engineering, SE-17,N 3, March 1991.

[37] Ben-Ari M., Manna, Z., Pnueli A.: The Temporal Logic of Branching Time. Proc. 8th Annual Symposium on Principles of Programming Languages, 1981, ACM Press, Williamsburg, p. 164-176, Springer-Verlag, 1992.

[38] Bouajjani A., Robbana R.: Verifying ω-regular properties for subclasses of linear hybrid systems. CAV, Springer LNCS, 1995.

[39] Burch J.R., Ckarke E.M., McMillan K.L., Dill D.L., Hwang L.J.: Symbolic model checking: 10^20 states and beyond. Informations and Computation, 98(2), p. 142–170, 1992.

[40] Bryant R.E.: Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers, C-35(8), p.677–691, 1986.

[41] Clarke E.M., Emerson E.A.: Design and synthesis of synchronisation skeletons using branching-time temporal logic. In Workshop on Logic of Program, Lecture Notes in Comp.Sc. 131. Springer-Verlag, 1981.

[42] Clarke E.M., Emerson E.A., Systla A.P.: Automatic verification of finite-state concurrent systems using temporal-logic specifications. ACM Transactions on Programming Languages and Systems, 8(2): p.244-263. 1986.

[43] Emerson E.A., Clarke E.M.: Characterizing correctness properties of parallel programs as fixpoints. In ICALP 83: Automata, Languages and Programming, LNCS 85, p.169–181, Springer-Verlag, 1980.

[44] Harel D.: Statecharts: a Visual Formalism for Complex Systems. Sci. Comput. Prog. 8, p.231-274, 1987.

[45] Henzinger T.A., Manna, Z., Pnueli A.: Temporal proof methodologies for real-time systems. Proc. 18th Annual Symposium on Principles of Programming Languages, 1991, ACM Press, p. 353-366.

[46] Henzinger T., Manna Z., Pnueli A.: Towards Refining Temporal Specifications into Hybrid Systems. In Hybrid Systems, LNCS 736, p.60–76, Springer-Verlag, 1993.

[47] Henzinger T., Nicollin X., Sifalis J., Yovine S.: Symbolic Model-Checking for Real-Time Systems. In Proc. 7th LICS. IEEE Comp. Soc. Press, 1992.

[48] Henzinger T., Ho P-T.: HyTech: The Cornell Hybrid Technology Tool. Hybrid Systems II, Lecture Notes in Comp.Sci 999, p.265-293. Springer-Verlag, 1995.

[49] Henzinger T.A., Ho P.-H.: Algorithmic analysis of nonlinear hybrid systems. CAV 95, Lecture Notes in Comp. Sci. p. 225–238. Springer-Verlag, 1995.

[50] Henzinger T.A., Wong-Toi H.: Using HyTech to synthesize control parameters for a steam boiler. Lecture Notes in Comp. Sci. 1165, p. 265–282. Springer-Verlag, 1996.

[51] Henzinger T.A., Ho P.-H., Wong-Toi H.: HyTech : a model checker for hybrid systems. Lecture Notes in Comp. Sci. 1254, p. 460–463. Springer-Verlag, 1997.

[52] T.A. Henzinger: Hybrid Automata with Finite Bisimulations. Lecture Notes in Comp. Sci. 944, p. 324–335. Springer-Verlag, 1995.

[53] Henzinger T.A., Kopke P., Puri A., Varaiya P.: What's decidable about hybrid automata? ACM Symp. of Theory of Computing, p.373–382, 1995.

[54] Ho P.-H.: Automatic Analysis of Hybrid Systems. PhD thesis, Cornell University, 1995.

[55] Inichova M.A., Inichov D.B., Kolesov Y.B., Senichenkov Y.B.: Movel Vision for Windows. Graphical environment for hybrid systwr simulating. User's Guade. Moscow-St.Petersburg, 1995.

[56] Kesten Y., Pnueli A., Sifalis Y., Yovine S.: Integration Graph: a class of decidable hybrid systems. Hybrid Systems, Lecture Notes in Comp.Sci 736, p.179-208. Springer-Verlag, 1993.

[57] Kolesov Y.B., Senichenkov Y.B.: Model Vision 3.0 for Windows 95/NT. The graphical environment for complex dynamic system design. ICI&C'97 PROCEEDINGS, v.2, p.704-711, St.Petersburg, 1997.

[58] Kolesov Y.B., Senichenkov Y.B.: Visual specification language intended for event-driven hierarchical dynamic system with variable structure. ICI&C'97 PROCEEDINGS, v.2, p.712-719, St.Petersburg, 1997.

[59] Lamport L.: Specifying concurrent program modules. ACM Trans. Prog. Lang. Syst. 5, p.190–222, 1983.

[60] Lamport L.: What good is temporal logic. Proc.IFIP Congress,North-Holland, p.657–668, 1983.

[61] Larsen K., Pettersson P., Yi W.: Compositional and symbolic model-checking of real-time systems. in Proceeding of the 16th Annual Real-Time Systems Symposium, p.76–87, IEEE Comp.Sc.Press, 1995.

[62] Maler O., Manna Z., Pnueli A.: From Timed to Hybrid systems. Real-Time: Theory in Practice, Lecture Notes in Comp.Sc 600, p.447-484. Springer-Verlag, 1992.

[63] Manna Z., Pnueli A.: Verification of concurrent programs: A temporal proof system. Foundation of Comp.Sc 4, Distributed Systems: Part 2, Mathematical Centre Tracts 159, Center for Mathematics and Computer Science, Amsterdam, p.163–255, 1983.

[64] Manna, Z., Pnueli A.: The Temporal Logic of Reactive and Concurrent Systems. Springer-Verlag, 1992.

[65] Manna Z., Pnueli A.: Verifying hybrid systems. Hybrid systems, LNCS 736, p.4–35, Springer-Verlag, 1993.

[66] McManis J., Varajya P.: Suspension Automata: A Decidable Class of Hybrid Automata. 6^{th} International Conference CAV'94. Lecture Notes in Comp. Sci. 818, p.105–117, 1994.

[67] Nicollin X., Olivero A., Sifalis Y., Yovine S.: An Approach to the Description and Analysis of Hybrid Systems. Hybrid Systems, Lecture Notes in Comp.Sci 736, p.149-178. Springer-Verlag, 1993.

[68] Olivero A., Yovine S.: Kronos: a tool for verifying real-time systems. User's Guide and Reference Manual. VERIMAG, Grenoble, France, 1992.

[69] Olivero A., Sifakis J., Yovine S.: Using Abstractions for the Verification of Linear Hybrid Systems. 6^th International Conference CAV'94. Lecture Notes in Comp. Sci. 818, p.81–94, 1994.

[70] Pnueli A., Rosner R. On the synthesis of a reactive module. Technical report, Weizmann Institute of Dcience, 1988.

[71] Puri A., Varajya P.: Decidability of Hybrid Systems with Rectangular Differential Inclusions. 6^th International Conference CAV'94. Lecture Notes in Comp. Sci. 818, p.95–104, 1994.

[72] Stauner T., Muller O., Fuchs M.: Using HyTech to verify an automotive control system. Lecture Notes in Comp. Sci. 1201, p. 139–153. Springer-Verlag, 1997.

[73] Wong-Toi H.: Symbolic approximations for verifying real-time systems. PhD thesis, Standford University, 1994.

[74] Wolfram S.: Mathematica: A System for Doing Mathematics by Computer. Addisson–Wesley Publishing Company, 1988.

[75] Redfern D.: The Maple Handbook. 531 pp., Springer-Verlag, 1994.

[76] SIMULINK. The ultimate simulation environment. MathWorks, 1994.

Приложение 1. Применение символьного подхода в задачах моделирования и анализа непрерывно–дискретных систем

В разное время предлагались различные математические модели непрерывно–дискретных систем, разрабатывались специальные формализмы описания, языки и системы моделирования. Среди них — кусочно–линейные системы А.А.Андронова (1953) [1], агрегативные системы Н.П.Бусленко (1963) [8], язык схем в переменных состояния Ю.Ту (1964), непрерывно–дискретные системы (и система моделирования НЕДИС) В.М.Глушкова (1973) [19], гибридные системы А.Пнуэли (1992), которым посвящена настоящая работа, и другие.

В работе автора [23] был проведен сравнительный анализ агрегативной системы, непрерывно-дискретной системы, кусочно-линейной системы и гибридного автомата.

Основным результатом анализа явилось установление частичного взаимного соответствия между моделями, которое изображено на рис.10. Несмотря на то, что каждый из авторов (или групп авторов), предлагая свою модель и новый формализм описания, предполагал, что определяет новый класс сложных систем, рассмотренные математические модели представляют собой различные подходы к описанию, моделированию и анализу одного класса сложных динамических систем (которые мы называем непрерывно-дискретными и поведение которых представлено на рис.1).

В работе [23] рассматривается специфика и ограничения каждой модели, которые определяются спецификой соответствующих подходов к исследованию непрерывно–дискретных систем — качественный анализ автоколебательных движений в кусочно–линейных системах, непрерывное численное моделирование по принципу "особых состояний" в агрегативном подходе с использованием кусочно–разностных методов, дискретный событийный подход, реализованный в системе НЕДИС В.М.Глушкова.

Перечислим кратко особенности каждой модели:

1. В кусочно-линейной системе невозможна иерархия элементов и смена структуры системы в смысле динамического изменения в со-

Рис. 10: Математические модели непрерывно-дискретных систем.

ставе ее элементов (хотя формально изменение размерности фазового пространства в областях линейности системы может быть интерпретировано как смена структуры). Специфика данного класса определяется локальными поведениями, описываемыми линейными дифференциальными уравнениями, и типом разрывов, описываемых условиями скачка с обязательным учетом предыстории (полный разрыв невозможен).

2. В агрегативной системе не предусматривается возможность смены структуры системы, другим ограничением является конечный интервал моделирования. Специфика агрегативного подхода заключается в использовании главным образом случайных последовательностей в качестве последовательностей приема входных, управляющих сигналов, оператора H; основной областью использования агрегативного подхода является моделирование систем массового обслуживания. Кроме того, формальное определение агрегата и определение агрегативной системы общего вида, предложенное Н.П.Бусленко, оказывается значительно шире того класса систем, который может быть реально смоделирован в вычислительных системах агрегативного направления, поскольку в них используются методы, базирующиеся на конечно–разностных уравнениях. Фактически задачу моделирования можно считать решенной для класса кусочно–линейных агрегатов.

3. В модели В.М.Глушкова не предусматривается возможность иерархии, кроме того, также как и в агрегативном подходе, имеется предположение о конечности интервала моделирования (считается, что календарь событий становится в некоторый момент пустым). Порождение и удаление процессов является механизмом смены структуры системы. Специфика модели В.М.Глушкова заключается в близости ее к дискретным моделям параллельных и распределенных систем. В ней удобно представлять чисто дискретные процессы и чисто дискретные периоды поведения системы.

4. В гибридной системе не предусматривается возможность смены структуры. Специфика гибридной модели заключается в возможности качественного и параметрического анализа систем, в которых значения параметров задаются в виде интервалов. Другая отличительная черта гибридной модели заключается в ее реактивности (подробнее см.[23]).

Описание математических моделей в гибридном подходе

1. Кусочно–линейная система

Следуя классификации нелинейных динамических систем, предложенной А.А.Андроновым, А.А.Виттом и С.Э.Хайкиным в теории нелинейных колебаний [1], автоколебательные системы, являющиеся подклассом нелинейных неконсервативных динамических систем, включают в себя три основных класса:

1) автоколебания, близкие к гармоническому осциллятору (то есть близкие к линейным консервативным системам), не имеющие разрывов в фазовых траекториях,

2) класс кусочно–линейных систем, который подразделяется на системы с разрывами в фазовых траекториях и системы без разрывов, для которых характерно "склеивание по непрерывности" фазовых траекторий,

3) "разрывные колебания", включающие в себя системы с разрывами вследствие наличия в уравнениях движения малого параметра и системы с разрывной правой частью в уравнениях движения.

Два последних класса обладают двумя важными свойствами, ха-

рактерными для непрерывно–дискретных систем, а именно мгновенной сменой поведения и наличием разрывов в фазовых траекториях. Рассмотрим класс кусочно–линейных систем [1],[15].

Определение 1. *Кусочно–линейной системой называется динамическая система, фазовое пространство которой состоит из областей, в которых динамические уравнения движения линейны.*

Описание кусочно–линейной системы включает в себя:

- описание областей фазового пространства $S_1, \ldots, S_n$ ("областей линейности") в виде линейных ограничений на значения отдельных фазовых переменных системы;
- конечный набор систем линейных дифференциальных уравнений $\{F_1, \ldots, F_n\}$, описывающих локальные поведения системы в соответствующих областях фазового пространства;
- описание "скачков" при переходе фазовых траекторий из одной области в другую вида $X_{\text{после}} = f_i(X_{\text{до}}), X_{\text{до}} \in bound(S_i)$, где X – вектор фазовых переменных, $f_i : R^n \to R^n$ – вещественная функция, через $bound$ обозначена граница области (при $f_i \equiv 1$ имеем "склеивание по непрерывности");
- описание начального состояния системы X_0.

Классическими примерами кусочно–линейных систем являются модели с мгновенными ударами (задачи с часовыми ходами и пр.)[5], модель лампового генератора, задача "о двухпозиционном рулевом" [9] и многие другие.

Опишем кусочно-линейную систему в терминах математической модели гибридного подхода. Определим гибридный автомат $H = \{V, E, X, \Sigma, flow, pre, post, init, inv, syn\}$, в котором каждой области S_i соответствует своя локация $v_i \in V$, множество дуг $E = \{e = (v', v''), \forall v', v'' \in V\}$ включает в себя множество всевозможных пар вершин автомата, а множество фазовых переменных X суть множество переменных кусочно–линейной системы. Остальные компоненты описания определим следующим образом:

$flow = \{F_1, \ldots, F_n\}$ – набор систем линейных дифференциальных уравнений, описывающих локальные поведения кусочно–линейной системы в областях линейности $S_1, \ldots, S_n$ фазового пространства систе-

мы;

$pre = \{X(t) = bound(S_i), i = 1 \dots n\}$ – набор уравнений, описывающих условия пересечения фазовой траектории $X(t)$ границ областей линейности фазового пространства кусочно–линейной системы;

$post = \{X_{\text{после}} = f_i(X_{\text{до}}), X_{\text{до}} \in bound(S_i)$ – множество инициализаций суть множество описаний допустимых скачков при переходах фазовой траектории кусочно–линейной системы из одной области линейности в другую;

$inv = \{S_1, \dots, S_n\}$ – описание областей фазового пространства $S_1, \dots, S_n$;

$init = \{X_0\}$; $\Sigma = \emptyset, syn = \emptyset$.

Каждая локация автомата имеет инвариант $inv(v_i) = S_i$. Каждая дуга $e = (v_i, v_j)$ помечена условием перехода $pre(e)$ вида $bound(S_i) \wedge bound(S_j)$ и инициализацией на дуге $post(e)$ вида $X_{\text{после}} = f_i(X_{\text{до}}) \wedge X_{\text{после}} \in S_j$.

Получим, таким образом, гибридный автомат с линейными системами ОДУ. Задание множества начальных состояний точкой фазового пространства и задание множества мгновенных действий *post* описанием условий скачков кусочно-линейной системы предопределяет наличие у автомата свойства детерминированной инициализации.

Таким образом, *произвольная кусочно–линейная динамическая система может быть описана моделью гибридного автомата с линейными системами ОДУ и с детерминированной инициализацией.*

Круг вопросов, который ставится в теории колебаний для кусочно–линейных систем, включает в себя качественное исследование периодических движений, их устойчивости, зависимость характеристик периодических движений (и их наличия) от параметров системы. Все эти вопросы могут быть исследованы символьными методами верификации.

2. Агрегативная система Н.П.Бусленко

Рассмотрим математическую модель агрегативной системы, предложенной Н.П.Бусленко в 1963 году [8] для моделирования сложных систем управления.

Определение 2. *Агрегатом называется следующая математическая модель:*

$A = \{ T, Z, X, \Gamma, Y, H, G \}$, *где*

$T = [0, T_f] \subset R$ — *интервал моделирования (обычно конечный);*

Z — *множество состояний (фазовое пространство);*

$z = (z_1, z_2, \ldots, z_l)$, *где* $z_1, \ldots z_l$ — *фазовые координаты;*

$\{z(t)\}$ — *фазовые траектории;*

$\{z(0)\}$ — *множество начальных состояний;*

X — *множество входных сигналов (*$x = [x^{(1)}, \ldots, x^{(l)}]$, *где* l *– число "входных контактов", каждый из которых принимает сигналы своего типа); моменты прихода входных сигналов определяются через временную последовательность* $\{t_j\}, t_j \in T$; *последовательность событий* $< t_j, x_j >, x_j \in X$ *называется последовательностью приема входных сигналов;*

Γ — *множество управляющих (особенных) сигналов (*$g = [g^{(1)}, \ldots, g^{(r)}]$, *где* r *– число "особенных входных контактов"); моменты прихода управляющих сигналов определяются через временную последовательность* $\{\tau_i\}, \tau_i \in T$; *последовательность событий* $< \tau_i, g_i >, g_i \in \Gamma$ *называется последовательностью приема управляющих сигналов;*

Y — *множество выходных сигналов; моменты выдачи выходных сигналов определяются через временную последовательность* $\{\xi_k\}, \xi_k \in T$, *в которой каждый элемент* ξ_k *соответствует моменту пересечения фазовой траектории* $z(t)$ *границы некоторого множества из системы множеств* $\{Z_y\}$, *определенной на пространстве состояний* Z *заданием набора предикатов над* Z; *последовательность событий* $< \xi_k, y_k >, y_i \in Y$ *называется последовательностью выдачи выходных сигналов;*

$H = \{V_x, V_g, V_{gx}, W, U\}$ *– оператор переходов, который определяет текущее состояние по предыстории:* $z(t) = H[z(0), t]$;

V_x, V_g, V_{gx} — *операторы формирования новых начальных условий при приеме очередного входного, управляющего сигналов и одновременного их приема соответственно (*V_g *формирует новое поведение,* V_x *формирует только новые значения параметров);* W — *оператор формирования новых начальных условий (как и* V_x, *только значений*

параметров) в момент выдачи выходного сигнала; U — оператор поведения на временных интервалах между событиями;

$G = G_? \cup G_!$ — оператор выходов, $G_?$ — оператор проверки условия выдачи выходного сигнала, $G_!$ — генератор выходного сигнала: $y(t) = G[z(0), t]$.

Состояние агрегата в момент получения входных и управляющих сигналов или выдачи выходного сигнала называется "особым", в эти моменты состояние агрегата может измениться скачкообразно.

Модель агрегата может быть использована как модель всей непрерывно–дискретной системы или ее элемента. В последнем случае система представляется сетью агрегатов с фиксированными каналами связей.

Определение 3. *Агрегативной системой называется любая совокупность агрегатов, если передача информации между ними происходит мгновенно и без искажений.*

Агрегативная система называется *комплексом*, если любой агрегат в ней соединен хотя бы с одним агрегатом системы. Агрегативная система называется *m–фазной*, если состоит из m последовательно соединенных комплексов. Агрегативная система называется *m–канальной*, если состоит из m параллельно соединенных комплексов. Агрегативная система называется *строго иерархичной*, если в ней можно выделить подчиненные и управляющие комплексы, то есть если существует (управляющий) комплекс, входная информация которого служит управляющей информацией другого (подчиненного) комплекса.

Моделирование поведения агрегата и агрегативной системы заключается в построении последовательности переходов из одного особого состояния в другое, причисляя к множеству особых состояний $z(0)$ (моделирование по принципу "особых состояний").

Конкретизация составных операторов оператора H приводит к выделению отдельных подклассов агрегативных систем. Например, кусочно-линейным агрегатом называется такой агрегат, у которого пространство состояний Z может быть разбито на счетное количество качественно различных ("основных") состояний, а соста-

вляющий оператор U оператора переходов H описывается кусочно-линейной функцией.

Определим подкласс агрегативных систем, который может быть описан в гибридном подходе.

Пусть дан агрегат $A = \{\ T,\ Z,\ X,\ \Gamma,\ Y,\ H,\ G\ \}$. Предположим, что его пространство состояний Z может быть определенным образом структрировано, а именно: в нем могут быть выделены конечное число "основных состояний" $v = 1, 2, \ldots, n$, соответствующих качественно различным состояниям агрегата и дополнительные координаты, которые характеризуют количественные изменения для каждого основного состояния (компонентами вектора дополнительных координат $z^{(v)}, z_i \in Z$ будут являться фазовые координаты, значимые для данного основного состояния v, их количество определяет ранг $\|v\|$ основного состояния). Предположим также, что пространство Z может быть разбито на n выпуклых многогранников

$$Z = \cup_v Z^{(v)}, dim Z^{(v)} = \|v\|$$

Множество начальных состояний агрегата $\{z(0)\}$ определим как некоторую (выбранную случайным образом) внутреннюю точку одного из многогранников.

Множества входных, управляющих и выходных сигналов представим аналогично с Z: $x = (\mu, x^{(\mu)}), g = (\nu, g^{(\nu)}), y = (\lambda, y^{(\lambda)})$, где μ, ν, λ — конечные множества целых чисел.

Определим подмножество $\{Z_y\}$, определяющее особые состояния, соответствующие выдачи выходных сигналов агрегата, таким образом, чтобы оно совпадало с гранями многогранников, и пусть составляющие операторы U и W оператора переходов H имеют вид:

$$U(v) : z(t) = \{^{v(t)=v=const}_{z^{(v)}(t)=f(v,t,z(v)(t_*+0))}$$

$$W(v) : z(t_* + 0) = (v, z^{(v)}(t_* + 0))$$

где f – векторная функция, заданная явно или являющаяся решением системы линейных дифференциальных уравнений, $z^{(v)}(t_*+0) \in Z^{(v)}$ — некоторая внутренняя точка (выбранная случайным образом) нового многогранника $Z^{(v)}$. Момент скачка t_* определим моментом пересечения траектории с ближайшей гранью многогранника текущего основного состояния.

Построим гибридный автомат $H = \{\ V,\ X,\ E,\ \Sigma,\ flow,\ pre,\ post,$

$inv,\ init,\ syn\ \}$, поведение которого будет соответствовать поведению агрегата A. Компоненты автомата определим следующим образом: каждая локация автомата $v_i \in V$ соответствует одному из основных состояний агрегата, $V = \{v_1, \ldots, v_n\}$;

$X = Z$ – фазовыми переменными автомата являются переменные агрегата;

$\Sigma = X \cup Y \cup \Gamma$ – объединим множества входных, управляющих и выходных сигналов в алфавит событий гибридного автомата;

$flow = \{f(v, t, z(v)(t_* + 0)), v \in V\}$ – локальные поведения в локациях соответствуют оператору U агрегата;

$pre = \{Z_y\}$ – множество предикатов включает в себя описание граней всех многогранников $Z^{(v)}$;

$post = \{V_x, V_{gx}, W\}$ – множество мгновенных действий при смене локальных поведений соответствует операторам W, V_x, V_{gx} агрегата и определяется как случайный выбор некоторой внутренней точки нового многогранника $Z^{(v)}$.

$inv = \{Z^{(v)}, v \in V\}$ – множество предикатов-инвариантов автомата является описанием многогранников;

$init = Z^{(v)},\ \ z(0) \in Z^{(v)}$ – множество начальных состояний автомата суть многогранник, к которому принадлежит начальное состояние $z(0)$ агрегата;

E – множество дуг автомата построим в соответствии с операторами агрегата $V = \{V_x, V_g, V_{gx}, W\}$, соединяя дугой каждые две вершины v и v' автомата, для которых в агрегате выполняется условие: $z(v', t_* + 0) = V[z(v, t_*)]$ (в том числе, в автомате будут присутствовать дуги вида (v, v), они будут соответствовать оператору V_x агрегата).

В каждую локацию автомата поместим соответствующий ей предикат-инвариант. Каждую дугу e пометим условием перехода $pre(e)$, инициализацией $post(e)$ и меткой перехода $syn(e)$ так, чтобы каждому сигналу агрегата отвечала соответствующая дуга автомата.

Таким образом, мы получили гибридный автомат с локальными поведениями, заданными функциями f, и с линейными предикатами.

В случае, если многогранники являются прямоугольными областя-

ми в R^n, а описания функций f агрегата являются системами линейных дифференциальных уравнений с постоянными коэффициентами, получим гибридный автомат с линейными системами ОДУ.

Автомат будет обладать свойством ограниченной интервальной инициализации, если область $Z^{(v)}$, соответствующая начальному состоянию $z(0)$, будет ограниченной.

Если все многогранники являются ограниченными прямоугольными областями в R^n, то построенный гибридный автомат будет иметь в каждой локации синхронизированный вход.

Если агрегат A является кусочно-линейным агрегатом, то построенный автомат будет линейным гибридным автоматом.

Параллельной композицией гибридных автоматов можно представить канальную агрегативную систему, если фазовое пространство каждого из составляющих ее агрегатов может быть разбито на конечное число основных состояний. При этом агрегаты могут иметь общие переменные и их множества входных, выходных и управляющих сигналов могут пересекаться.

Способ разбиения пространства состояний агрегатов на семейства $\{Z_y\}$, интерпретация операторов V_x, V_g, V_{gx}, W инициализациями на дугах автомата определяет взаимно однозначное соответствие между начальной точкой в вычислении гибридного автомата и особым состоянием агрегата $z(0)$, и между каждой другой точкой в вычислении гибридного автомата и особым состоянием агрегата, определяемым выдачей или приемом соответствующего сигнала. Следовательно, множество всевозможных последовательностей особых состояний, которые могут быть порождены агрегативной системой будут составлять поведение (множество линейных вычислений) гибридного автомата.

Таким образом, *произвольная канальная агрегативная система, фазовые пространства составляющих агрегатов которой могут быть разбиты на конечное число основных состояний, может быть описана параллельной композицией гибридных автоматов.*

3. Непрерывно–дискретная модель В.М.Глушкова

Рассмотрим математическую модель непрерывно–дискретной системы, предложенную В.М.Глушковым в 1973 году [19] (система моделирования НЕДИС).

Определение 4. *Непрерывно–дискретной системой называется следующая математическая модель:*

$$S = \{ T, P, e, E, K, F \}, \text{ где}$$

$T = \{t_i\}, t_i \in R_{\geq 0}$ — *дискретная модель времени;*

P — *множество классов процессов;*

e — *множество классов событий (причин мгновенной смены поведения и структуры системы);*

E — *множество алгоритмов классов событий (подготовительных дискретных действий при переходе к новому поведению системы). К элементарным действиям относятся пассивизация, активизация, порождение и уничтожение процессов, изменение значений переменных процесса и запись в календарь планирования событий отметки о будущем событии;*

$K = \{< t_i, e_i >, L\}$ — *календарь планирования событий, в который записываются отметки о событиях отдельными объектами и с помощью которого описывается динамика системы. Планирование события подразумевает явное задание момента его наступления или задание условия L его наступления через предикат (планирование события по условию);*

F — *список уравнений, описывающих локальные поведения процессов во временных интервалах между событиями.*

Структура процесса и его поведение описывается следующей математической моделью:

$$P = \{ X, Y, V_s, V_d, B \}, \text{ где}$$

X, Y — *каналы входа и выхода;*

V_s — *множество статических переменных процесса, значения которых задаются алгебраическими выражениями и могут меняться только при исполнении алгоритмов событий;*

V_d — *множество "переменных–функций" – динамических переменных, поведение которых описывается дифференциальными уравнени-*

ями из множества F;

B — тело процесса, содержащее описания его всевозможных поведений.

Под моделированием поведения непрерывно-дискретной системы понимается построение множества последовательностей событий, приводящих к смене ее поведения и структуры, причисляя к событию начальное состояние системы. Моделирование осуществляется специальным процессом-монитором, который продвигает системное время в соответствии с календарем планирования событий или в соответствии с анализом времени наступления событий, которые планируются по условию. Процесс моделирования заканчивается, когда календарь событий оказывается пустым.

Пусть непрерывно-дискретная система состоит из фиксированного числа процессов (операции порождения и удаления процессов запрещены). Зададим список уравнений F, описывающих локальные поведения процессов между событиями в виде конечного числа систем дифференциальных уравнений.

Рассмотрим отдельный процесс непрерывно-дискретной системы $P = \{\ X,\ Y,\ V_s,\ V_d,\ B\ \}$. Введем непрерывную модель времени. Обозначим через F_P подмножество локальных поведений F, отвечающих переменным-функциям процесса P. Обозначим через e_P подмножество классов событий e, которые влияют на изменение поведения процесса P, а через e_{LP} подмножество классов событий e, которые влияют на изменение поведения процесса P и планируются по условию L.

Построим гибридный автомат $H = \{\ V,\ E,\ X,\ \Sigma,\ flow,\ pre,\ post,\ inv,\ init,\ syn\ \}$, компоненты которого определим следующим образом: каждому локальному поведению $f \in F_P$ соответствует некоторая локация автомата $v_f \in V$; $X = V_s \cup V_d$; $\Sigma = e_P \cup e_{LP}$;

$flow = F_P \cup f_{idle}$ – к множеству локальных поведений процесса добавляем локальное поведение $f_{idle} : X = const$, описывающее поведение процесса в периоды пассивности;

$pre = L_P$ – множество условий планирования событий процесса P;

$post = E(e_P) \cup E(e_{LP})$ – мгновенные действия включают в себя алгоритмы событий процесса P;

$inv = \emptyset$ – дополнительных ограничений на значения фазовых переменных нет; $init = V_s(0) \cup V_d(0)$;

множество дуг автомата E определяется в соответствии с описанием поведения процесса, которое содержится в его теле B.

Построим далее параллельную композицию гибридных автоматов, в которой каждый автомат поставлен в соответствие некоторому процессу непрерывно-дискретной системы. Добавим к параллельной композиции таймированный автомат, поведение которого будет соответствовать процессу–монитору в непрерывно–дискретной системе.

Из данного построения видно, что множество дуг в построенной параллельной композиции гибридных автоматов включает в себя описание всех возможных классов событий, приводящих к смене поведения непрерывно–дискретной системы и событие начального запуска и не содержат никаких других сигналов. Каждому событию некоторого процесса непрерывно–дискретной системы соответствует единственная дуга соответствующего автомата, а каждой цепочке событий непрерывно–дискретной системы соответствует единственное линейное вычисление параллельной композиции автоматов. Следовательно, поведение непрерывно-дискретной системы описывается множеством вычислений построенной параллельной композицией гибридных автоматов. Задание начального состояния непрерывно-дискретной системы точкой фазового пространства и задание множества мгновенных действий *post* каждого автомата описанием дискретных действий соответствующего процесса непрерывно-дискретной системы предопределяет наличие у каждого автомата параллельной композиции свойства детерминированной инициализации.

Таким образом, *произвольная непрерывно–дискретная модель В.М.Глушкова с фиксированным числом процессов может быть описана параллельной композицией гибридных автоматов с детерминированной инициализацией.*

Если множество F непрерывно-дискретной системы задается линейными системами ОДУ с постоянными коэффициентами, то она может быть описана параллельной композицией гибридных автоматов с линейными системами ОДУ и детерминированной инициализацией.

Приложение 2. Пример. Метод покоординатных оценок для динамической модели часовых ходов.

Рассмотрим упрощенную идеальную модель часов, которая является неконсервативной динамической системой с двумя степенями свободы ([1], [5],[4]). Основными частями часовых ходов являются осциллятор (балансир или маятник), совершающий колебательное движение, и часовое колесо, приводимое в движение пружиной и передающее импульсы осциллятору. Примем следующие упрощающие предположения: а) импульс передается мгновенно, в момент соударения ходового колеса с балансиром; б) кинетическая энергия балансира рассматривается только за счет сил вязкого трения и ударов; в) пренебрегаем всеми нелинейностями системы, порождаемыми кинематическими связя

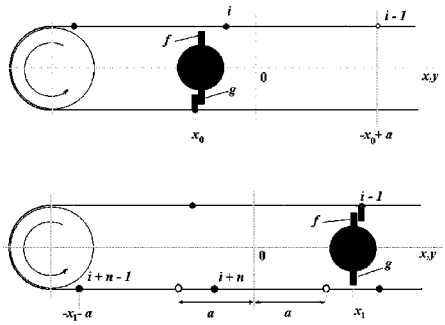


Рис. 11: Два последовательных послеимпульсных состояния динамической модели часовых ходов.

В качестве модели часового хода часто используют механическую систему (рис.11), в которой ходовое колесо представляется в виде бесконечной ленты, перекинутой через блок и снабженной зубцами, имитирующими зубцы ходового колеса. Балансир представляется как линейный осциллятор с горизонтальной осью, снабженный палетами f и g, которые воспринимают удары зубцов ленты. Считается, что в

промежутках между ударами лента имеет только один выдвинутый зубец, который после удара мгновенно убирается, а вместо него появляется новый по определенному закону. Кроме того, считается, что зубцы не служат препятствием для таких движений балансира, при которых он обгоняет зубцы ходового колеса. Зубцы располагаются на расстоянии $2a$ и последовательно пронумерованы. Удары по палетам f и g наносятся последовательно зубцами $i, i+n, i-1, i+n-1$ и т.д., причем положение равновесия балансира таково, что когда над положением равновесия находится зубец i, зубцы $i+n$ и $i+n+1$ снизу располагаются симметрично (на расстоянии a) относительно этого положения равновесия.

Положение балансира определим координатой x, положение произвольного зубца, передающего импульс, определим координатой y ($x = y = 0$ в положении равновесия).

В промежутках между ударами уравнения движения балансира и ходового колеса имеют вид:

$$m\ddot{x} + b\dot{x} + cx = 0, \quad M\ddot{y} = \pm P_0, \tag{32}$$

где m — приведенная масса балансира, b — коэффициент вязкого трения, c — коэффициент упругости пружины, M — приведенная масси ходового колеса, P_0 — сила, приводящая во вращение ходовое колесо, "$\pm$" — знаки для верхних и нижних зубцов соответственно.

Обозначив через $\{\dot{x},\ \dot{y}\}$ и $\{\dot{x}',\ \dot{y}'\}$ — доударные и послеударные скорости, опишем удары сотношениями (для не вполне упругого удара):

$$\begin{cases} \dot{x}' = \alpha\dot{x} + (1-\alpha)\dot{y} \\ \dot{y}' = \beta\dot{x} + (1-\beta)\dot{y} \end{cases} \tag{33}$$

где $\alpha = (J_0 - k)/(1 + J_0),\ \beta = \alpha + k, \quad J_0 = m/M, \quad k$ — коэффициент восстановления не вполне упругого удара.

Задача часовых ходов является примером кусочно–линейной системы. Классическое решение ее сводится к построению точечных преобразований для переходов между послеударных состояний, порождаемыми последовательными ударами о палеты f и g, и нахождению его неподвижных точек в фазовом пространстве размерности 3:

Рис. 12: Карта состояний для модели часовых ходов в системе Model Vision 3.0.

$\{x(=y),\ \dot{x},\ \dot{y}\}$. В силу симметричности фазового пространства (уравнения (32), (33) не меняются при замене $x, y, \dot{x}, \dot{y}$ на $-x, -y, -\dot{x}, -\dot{y}$), рассмотрение последовательности послеударных состояний сводится к рассмотрению преобразования пространства $\{x, \dot{x}, \dot{y}\}(\dot{x} - \dot{y} \geq 0)$ в себя при неограниченном повторении преобразования $T = \Omega^*_{gf}$ ($*$ означает переход Ω_{gf} в симметричную точку). Построение преобразования Ω_{gf} подробно рассмотрено в [5].

Автоколебания в системе часовых ходов могут быть исследованы также в компьютерных системах моделирования, базирующихся на численных методах.

С помощью системы компьютерного моделирования Model Vision 3.0 [57] мы можем подобрать значения параметров и начальные значения переменных системы, соответствующие автоколебаниям:

$$a = 0.2, b = 0.4, c = 0.2, P_0 = 0.2, M = 1, m = 1, \alpha = 0.5, \beta = 0.5,$$
$$x_0 = -0.043, y_0 = 0.243, \dot{x}_0 = 0.110, \dot{y}_0 = -0.13.$$

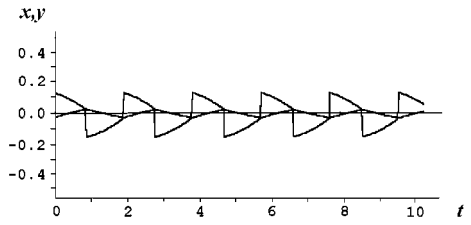


Рис. 13: Автоколебания в системе часовых ходов (Model Vision 3.0).

Карта состояний, построенная для модели часовых ходов, изображена на рис.12, автоколебания с нашем случае будут иметь вид, изображенный на графике 13, период установившихся колебаний равен $T = 1.8$сек.

Построим гибридный автомат для системы часовых ходов.

Пусть $p = P_0/2M$, $h = b/2m$, $k = c/m$. Гибридный автомат (рис.14) будет иметь две локации, соответствующие двум полупериодам автоколебательного движения системы, уравнения движения в локациях имеют вид:

$$\text{diff}(1) = \begin{cases} \ddot{x} + 2h\dot{x} + kx = 0 \\ \ddot{y} = -2p \end{cases} \tag{34}$$

$$\text{diff}(2) = \begin{cases} \ddot{x} + 2h\dot{x} + kx = 0 \\ \ddot{y} = 2p \end{cases}$$

Условия переходов по дугам автомата, очевидно, определяются ударом балансира и палеты:

$$pre(1) \;=\; pre(2) \quad : \quad x = y \tag{35}$$

Описание удара (33) соответствует инициализации на дугах автомата:

$$post(e_{12}) = \begin{cases} \dot{x}' = \alpha\dot{x} + (1-\alpha)\dot{y} \\ \dot{y}' = -(\beta\dot{x} + (1-\beta)\dot{y}) \\ y' = -x - a \end{cases} \quad post(e_{21}) = \begin{cases} \dot{x}' = \alpha\dot{x} + (1-\alpha)\dot{y} \\ \dot{y}' = -(\beta\dot{x} + (1-\beta)\dot{y}) \\ y' = -x + a \end{cases}$$

Наконец, начальные условия $x = x_0, y = -x_0 + a, \dot{x} = \dot{x}_0, \dot{y} = \dot{y}_0$, описывающие состояние системы непосредственно после удара о палету g, определяют начальный предикат init(1).

Гибридный автомат имеет единственный несинхронизированный цикл, поэтому в соответствии с теоремой 2.3.7 к нему могут быть применен метод символьной верификации, если области всех возможных начальных значений локаций 1 и 2 не разрастаются. Для выяснения этого обстоятельства достаточно решить задачу оценивания фазового состояния последовательно для каждой локации гибридного автомата.

Продемонстрируем решение последовательности задач оценивания для локации 1.

Решение системы дифференциальных уравнений (34) известно (см. [5]):

$$\begin{cases} x = \frac{\mu x_0 - \dot{x}_0}{\mu - \lambda} e^{\lambda t} + \frac{\lambda x_0 - \dot{x}_0}{\lambda - \mu} e^{\mu t}, \\ \dot{x} = (\frac{\mu\lambda}{\mu - \lambda} x_0 - \frac{\lambda}{\mu - \lambda} \dot{x}_0) e^{\lambda t} + (\frac{\mu\lambda}{\lambda - \mu} x_0 - \frac{\mu}{\lambda - \mu} \dot{x}_0) e^{\mu t}, \\ y = -pt^2 + \dot{y}_0 t + y_0, \\ \dot{y}' = -2pt + \dot{y}_0 \end{cases} \tag{36}$$

где μ, λ — корни (действительные или комплексные) характеристического уравнения $\xi^2 + 2h\xi + k = 0$. Поэтому решение задачи оценивания упрощается.

Используем значения параметров модели, соответствующие автоколебательному режиму, найденному компьютерным моделированием в Model Vision:

$$a = 0.2, h = 0.2, k = 0.2, p = 0.1, \alpha = 0.5, \beta = 0.5,$$

Начальные значения зададим в виде интервалов:

$x_0 = [-0.055, -0.035], y_0 = [0.235, 0.255],$

$$\dot{x}_0 = [0.105, 0.135], \dot{y}_0 = [-0.135, -0.105]. \tag{37}$$

Корни характеристического уравнения $\xi^2 + 0.4\xi + 0.2 = 0$ комплексно-сопряженные и равны $\mu, \lambda = -0.2 \pm 0.4i$. Для комплексно-сопряженных корней $\lambda_{1,2} = -h \pm i\omega$ компоненты решения системы $x, \dot{x}$

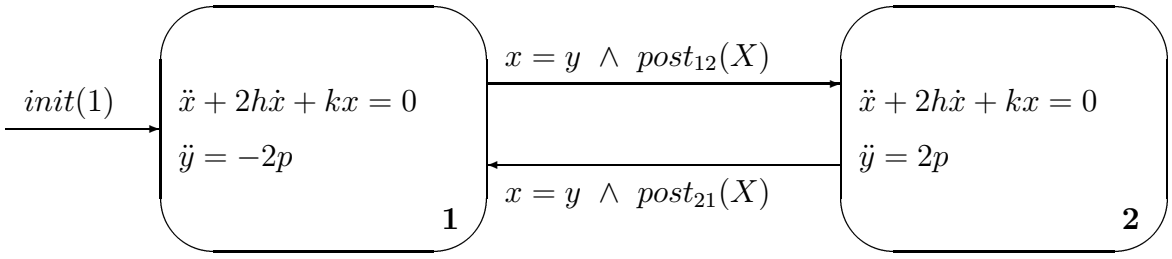


Рис. 14: Гибридный автомат динамической модели часовых ходов.

принимают вид:

$$\begin{cases} x = e^{-ht}(x_0 cos\omega t + \frac{\dot{x}_0 + hx_0}{\omega} sin\omega t), \\ \dot{x} = e^{-ht}(\dot{x}_0 cos\omega t - \frac{kx_0 + h\dot{x}_0}{\omega} sin\omega t) \end{cases} \tag{38}$$

где $\omega = \sqrt{k - h^2}$.

Предикат (35) является частным случаем предиката $\sum_{i=1}^{n} h_i x_i = c$ и описывает гиперплоскость в пространстве состояний системы, следовательно, мы имеем вырожденный случай прохождения пучка решений задачи Коши через гиперплоскость в фазовом пространстве. Перебираем вершины прямоугольной области в R^4 (37) и находим последовательно первые корни уравнений $x(t) - y(t) = 0$, так как $\mathbf{x}_0$ не принадлежит гиперплоскости. Получим множество значений t и соответствующее ему множество точек пересечения точечных решений задачи Коши и гиперплоскости.

Выбрав из множеств максимальные и минимальные значения, будем иметь аппроксимацию искомой области прямоугольным параллелепипедом $X_{\text{вых}}(1)$ в R^4:

$x_{\text{вых}} = [0.033, 0.058], \quad \dot{x}_{\text{вых}} = [0.074, 0.090],$

$\dot{y}_{\text{вых}} = [-0.332, -0.278], \quad y_{\text{вых}} = x_{\text{вых}}$

Выбрав из множества значений t минимального и максимального значения получим искомый интервал времени $t_{1\to 2} = [0.84, 1.01]$.

Вычислим область начальных значений в локации 2:

$$X_0(2) = post(e_{12})(X_{\text{вых}}(1)) = \begin{cases} \dot{x}' = \alpha\dot{x} + (1-\alpha)\dot{y} = [-0.121, -0.102] \\ \dot{y}' = -\beta\dot{x} - (1-\beta)\dot{y} = [0.102, 0.121] \\ x' = x = [0.033, 0.058] \\ y' = -x - a = [-0.258, -0.233] \end{cases}$$

Решив задачу оценивания для локации 2, получим $t_{2\to1}$ и $X_{\text{вых}}(2)$:
$t_{2\to1} = [0.90, 1.058]$,
$x_{\text{вых}} = [-0.058, -0.0349]$, $\dot{x}_{\text{вых}} = [0.067, 0.082]$,
$\dot{y}_{\text{вых}} = [-0.33, -0.28]$, $y_{\text{вых}} = x_{\text{вых}}$
Следовательно,

$$X_0(1) = post(e_{21})(X_{\text{вых}}(2)) = \begin{cases} \dot{x}' = \alpha\dot{x} + (1-\alpha)\dot{y} = [0.108, 0.131] \\ \dot{y}' = -\beta\dot{x} - (1-\beta)\dot{y} = [-0.131, -0.108] \\ x' = [-0.058, -0.0349] \\ y' = [0.235, 0.258] \end{cases} \tag{39}$$

Сравнивая входные значения переменных системы до и после прохождения по несинхронизированному циклу (формулы (37) и (39)), делаем вывод о неразрастании области входных значений для локации 1. Аналогичным образом можно показать неразрастание области входных значений для локации 2. Таким образом, модель часовых ходов с начальными значениями (37) и значениями параметров системы удовлетворяет условиям теоремы 2.3.7 и следовательно, к ней могут быть применены методы символьной верификации гибридных систем.

Продемонстрируем теперь метод оценки временного интервала для локации 1. В интервальном виде решение (38) будет следующим:

$$\begin{cases} x = e^{-0.2t}([-0.055, -0.035]cos0.4t + [0.257, 0.297]sin0.4t), \\ \dot{x} = e^{-0.2t}([0.105, 0.135]cos0.4t - [0.0275, 0.0475]sin0.4t), \\ y = -0.1t^2 + [-0.135, -0.105]t + [0.235, 0.255], \\ \dot{y}' = -0.2t + [-0.135, -0.105] \end{cases} \tag{40}$$

Подставляем полученные покомнонентные решения (40) в интервально–значную функцию $f(t)$, которая соответствует предикату (35),

описывающему условие перехода из локации 1 в локацию 2:

$$f(t) = x - y = e^{-0.2t}(\ [-0.055, -0.035]cos0.4t + [0.257, 0.297]sin0.4t\ )$$

$$+\ 0.1t^2 + [0.105, 0.135]t - [0.235, 0.255]$$

Находим граничные функции $f_1(t), f_2(t)$ для $f(t)$ в области $t > 0$ в соответствии с формулой (30):

$$f_1(t) = e^{-0.2t}[\ (-0.055\theta_{cos} - 0.035\gamma_{cos}\ )\ cos0.4t +$$

$$+\ (\ 0.232\theta_{sin} + 0.284\gamma_{sin}\ )\ sin0.4t\ ] +\ 0.1t^2 + 0.105t - 0.255$$

$$f_2(t) = e^{-0.2t}[\ (-0.055\gamma_{cos} - 0.035\theta_{cos}\ )\ cos0.4t +$$

$$+\ (\ 0.232\gamma_{sin} + 0.284\theta_{sin}\ )\ sin0.4t\ ] +\ 0.1t^2 + 0.135t - 0.235$$

где $\theta_{cos}, \gamma_{cos}, \theta_{sin}, \gamma_{sin}$ вычисляются по формулам:

$$\theta_{cos} = \frac{1 + sgn(cos\ 0.4t)}{2} = \begin{cases} 1, & \text{если } cos\ 0.4t > 0 \\ 0, & \text{если } cos\ 0.4t < 0 \end{cases}$$

$$\gamma_{cos} = 1 - \theta_{cos} = \begin{cases} 0, & \text{если } cos\ 0.4t > 0 \\ 1, & \text{если } cos\ 0.4t < 0 \end{cases}$$

$$\theta_{sin} = \frac{1 + sgn(sin\ 0.4t)}{2} = \begin{cases} 1, & \text{если } sin\ 0.4t > 0 \\ 0, & \text{если } sin\ 0.4t < 0 \end{cases}$$

$$\gamma_{sin} = 1 - \theta_{sin} = \begin{cases} 0, & \text{если } sin\ 0.4t > 0 \\ 1, & \text{если } sin\ 0.4t < 0 \end{cases}$$

Для вычисления искомого интервала времени необходимо найти первый положительный корень уравнения $f_1(t) = 0$ и первый положительный корень уравнения $f_2(t) = 0$. Результат вычислений дает значения $t_1 = 0.841,\ t_2 = 1.010$. На рисунке 15 изображена интервально–значная функция $f(t)$ и интервал времени, когда она пересекает гитерплоскость $x - y = 0$.

Аналогично, можно вычислить интервал $t_{2\to1}$ (см.рис. 16) для локации 2 $t_{2\to1} = [0.90, 1.06]$.

Временные интервалы, полученные методом покоординатных оценок и методом оценки временного интервала, совпадают, это доказывает корректность метода оценки временного интервала.

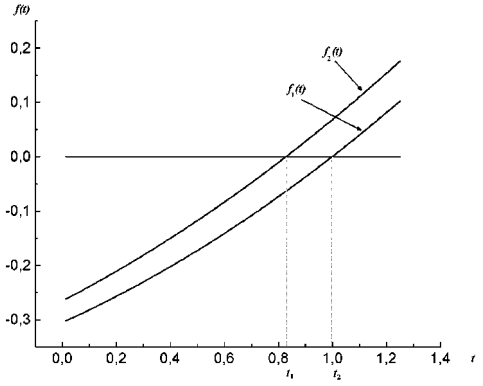


Рис. 15: Отыскание корней уравнения $f(t) = 0$ для локации 1 гибридного автомата часовых ходов.

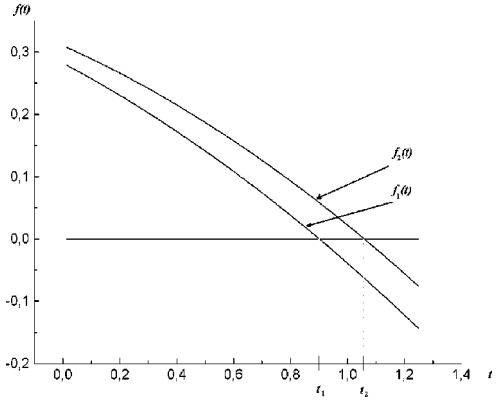


Рис. 16: Отыскание корней уравнения $f(t) = 0$ для локации 2 гибридного автомата часовых ходов.