

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

На правах рукописи

Васильев Павел Константинович

**РАЗРАБОТКА И РЕАЛИЗАЦИЯ СИСТЕМЫ
ИНТЕРПРЕТАЦИИ СПЕЦИФИКАЦИЙ НА
ЯЗЫКЕ ASM С ВРЕМЕНЕМ И ПРОВЕРКИ
СВОЙСТВ ТРАСС ИХ ВЫПОЛНЕНИЯ**

05.13.11 – Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата физико-математических наук

Санкт-Петербург – 2008

Работа выполнена на кафедре информатики математико-механического факультета Санкт-Петербургского государственного университета.

Научные руководители: кандидат физико-математических наук, доцент Соловьев Игорь Павлович, доктор наук, профессор Бокье Даниэль (университет Париж 12).

Официальные оппоненты: доктор физико-математических наук, профессор Баранов Сергей Николаевич (ЗАО «Моторола ЗАО»), кандидат физико-математических наук, доцент Кознов Дмитрий Владимирович (Санкт-Петербургский государственный университет).

Ведущая организация: Санкт-Петербургский институт информатики и автоматизации РАН.

Защита состоится «_____» _____ 2008 г. в _____ часов на заседании совета Д 212.232.51 по защите докторских и кандидатских диссертаций при Санкт-Петербургском государственном университете по адресу: 198504, Санкт-Петербург, Старый Петергоф, Университетский пр., д. 28, математико-механический факультет, ауд. 405.

С диссертацией можно ознакомиться в Научной библиотеке им. М. Горького Санкт-Петербургского государственного университета по адресу: 199034, Санкт-Петербург, Университетская наб., д. 7/9.

Автореферат разослан «_____» _____ 2008 г.

Ученый секретарь диссертационного совета,
доктор физико-математических наук,
профессор

Мартыненко Б.К.

Общая характеристика работы

Актуальность темы. История создания компьютерных систем всегда сопровождалась поиском средств точной спецификации программных и аппаратных продуктов. Жесткие требования, предъявляемые к качеству и срокам выхода продукта на современный рынок информационных технологий, заставляют специалистов постоянно работать над этой проблемой, пересматривать существующие подходы и предлагать новые решения проблемы в виде все более совершенных языков и систем спецификации, в том числе методов формальной спецификации, о которых преимущественно идет речь в данной работе.

Формализм *машин абстрактных состояний* (abstract state machines или ASM [17]) был введен в начале 1990-х годов Юрием Гуревичем. Изначально известные под именем *развивающиеся* или *эволюционирующие алгебры* (evolving algebras), машины Гуревича фактически стали одним из методов формальной спецификации компьютерных систем. Некоторые идеи метода ASM в виде отдельных концепций уже были известны на момент его создания: псевдокод, концепции виртуальных машин компании IBM и абстрактных машин Дейкстры [13], а также структуры Тарского [20], как наиболее общий метод представления абстрактных состояний некоторых вычислений. Метод ASM объединяет и развивает достоинства перечисленных концепций. С одной стороны, метод ASM является простым и понятным для пользователя, с другой, он позволяет разрабатывать крупномасштабные приложения. В качестве основных идей ASM включает в себя определение *локального замещения* абстрактного состояния некоторого вычисления (локальное изменение интерпретаций функциональных символов или, другими словами, переопределение значений интерпретирующих функций в отдельных точках), метод последовательного уточнения спецификации Н. Вирта [26] и концепцию базовой модели (ground model) [8], т. е. модель, полученную в результате формализации исходных требований пользователя.

В качестве множества состояний ASM выступают алгебраические структуры, а переходы между состояниями (эволюция алгебры) осуществляются с помощью операторов некоторого достаточно простого языка, синтаксис которого близок большинству известных процедурных языков программирования. Метод ASM позволяет применить технику последовательных уточнений спецификаций от одного уровня абстракции к другому. Такой подход помогает упростить проверку корректности спецификаций сложных программных систем, а операционный характер языка обеспечивает возможность перехода от формализованных описаний к выполнимым моделям. Выполнимость модели может быть использована для отладки высокоуровневых спецификаций и для создания тестовых примеров.

Язык ASM, который используется нами в качестве основы языка спецификации, обладает большой выразительной силой. Даже его подмножество, охватывающее *простейшие* ASM (Basic ASMs [18]), позволяет описать любую алгоритмическую машину состояний, включая машину Тьюринга. Методология ASM нашла свое применение в таких практических задачах, как спецификация семантики языков VHDL, C, C++, Prolog, C#, Java, SDL 2000, спецификация протоколов и других прикладных задач (см. [10, 24]). Та же методология была успешно применена в создании промышленных компьютерных систем, например, в проекте FALCO [9] компании Siemens (программное обеспечение для разработки и верификации расписаний железнодорожных линий). Еще одним примером применения машин Гуревича в промышленном масштабе является система AsmHugs [25], предназначенная для валидации технологий Microsoft, в частности модели COM. Основные достоинства метода ASM, которые упоминают разработчики, — это выполнимость спецификаций, возможность моделирования на различных уровнях абстракции, а также формальный переход от одного уровня абстракции к другому.

В настоящее время известен целый ряд реализаций интерпретаторов и компиляторов для различных диалектов языка ASM: Microsoft AsmL и Spec Explorer [5], XASM [4], CoreASM [15], Timed ASM (TASM) [21], Distributed ASML [23], ASM Workbench [11], ASM Gofer [22].

В процессе разработки сложной компьютерной системы очень часто возникает задача проверки ее корректности и адекватности. Известно, что намного выгоднее выявлять ошибки и противоречия еще на стадии спецификации системы при ее проектировании, так как этот этап предшествует этапу кодирования, объем спецификации обычно значительно меньше размера кода, а синтаксис языка спецификации проще и понятнее. Одним из часто используемых способов проверки корректности и работоспособности системы является тестирование. Тестирование состоит в многочисленных прогонах целевого кода на некотором конечном множестве входных данных с последующей проверкой полученного результата выполнения на соответствие существующим требованиям. Можно выделить еще один метод проверки, который часто называют *верификацией*. Он заключается в построении доказательства того, что, например, спецификация системы удовлетворяет начальным требованиям, т. е. доказывается, что на множестве всевозможных входных данных алгоритм заканчивает работу и вырабатывает удовлетворяющий требованиям результат. Поскольку это не всегда возможно, в данной работе под проверкой корректности свойств спецификаций подразумевается проверка заданных пользователем свойств на трассах выполнения спецификаций. Спецификация системы представляется в виде абстрактного высокоуровневого описания данных и алгоритмов. Пользовательские свойства можно разделить на описание состояния окружения и условия, которые должны выполняться после

завершения или во время работы алгоритма. Таким образом, задача проверки спецификации сводится к проверке того, что при определенных условиях окружения и удачном завершении работы алгоритма выполняются определенные пользователем ограничения.

Проектирование и реализация компьютерных систем с ограничениями на время реакции, которые рассматриваются в данной работе, подразумевают необходимость реализации подходящего механизма работы с временными значениями и входными данными. Исходный формализм ASM не предусматривает встроенной временной модели, однако существует ряд работ, предлагающих варианты решения этой задачи. Впервые решение проблемы предложили Гуревич и Хаггинс [19], которые представили вычисления в виде отображения из области временных значений в область состояний ASM. Затем последовали работы [7, 12], в которых проблема верификации временных алгоритмов сводится к верификации формальных спецификаций в виде формул специальной временной логики первого порядка FOTL (First Order Timed Logic).

Для описания свойств спецификаций в формализме ASM требуется достаточно выразительный язык логики. Самым простым логическим языком является язык пропозициональной логики. Однако, нетрудно заметить, что он плохо приспособлен для спецификаций систем с временем. Существует ряд расширений пропозициональной логики, такие, как логики с временем, или *темпоральные логики* [14], которые обладают большей выразительной способностью, например, LTL, CTL или их модификации. Такие логики широко используются для формальной спецификации программ и вычислительных систем. Хотя указанные логики и являются мощным инструментом формальной спецификации динамических систем, некоторые свойства, например, ограничения на время реакции системы, в них выразить очень сложно. В данной работе для спецификации свойств проектируемой системы применяется язык временной логики первого порядка FOTL. Логика FOTL была специально разработана Д. Бокье и А. Слисенко [6] как метод спецификации алгоритмов и их свойств с непрерывным временем. С помощью языка логики FOTL можно компактно специфицировать сложные свойства поведения компьютерных систем во времени.

В некоторых работах, посвященных использованию времени в методе ASM, предлагается явное использование функции времени, например, в [5, 16]. Кроме того, вводятся временные задержки на выполнение либо отдельных фрагментов вычислений, либо замещений [12, 21]. В перечисленных работах содержится множество идей и предложений по использованию и уточнению языка ASM, но, несмотря на это, в них не сформулирована явная методика спецификации систем с ограничениями на время реакции и проверки их свойств.

Из всего сказанного следует, что разработка и реализация специализированного расширения метода ASM для спецификации систем с временными ограничениями, а также разработка и реализация алгоритмов интерпретации спецификаций на расширенном языке и проверки ограничительных свойств являются актуальными задачами.

Цели и задачи диссертационной работы. Основными целями работы являются разработка расширения языка ASM, ориентированного на формальную спецификацию систем с ограничениями на время реакции, а также разработка и реализация программного инструмента для поддержки создания и выполнения таких спецификаций, а также проверки их свойств. В качестве языка спецификации ограничительных свойств в данной работе используется язык логики FOTL.

Для достижения обозначенных целей были поставлены следующие задачи:

- разработка временной модели для ASM, синтаксиса и семантики расширенного временем языка ASM;
- разработка метода проверки ограничительных свойств на языке FOTL применительно к спецификации на расширенном языке;
- разработка и реализация программной системы интерпретации расширенного языка ASM и проверки ограничительных свойств.

Основные результаты. Приведем основные результаты диссертационной работы:

- разработана временная модель ASM с непрерывным временем для спецификации компьютерных систем с ограничениями на время реакции;
- для предложенной временной модели разработаны синтаксическое расширение языка ASM, обеспечивающее адекватную и компактную запись спецификаций рассмотренного класса задач, а также семантика расширенного временем языка ASM;
- разработан синтаксис языка описания временных ограничений на базе языка временной логики первого порядка FOTL;
- разработаны алгоритмы решений уравнений для охраняющих условий, в которых используются внешние функции, имеющие кусочно-линейную зависимость от времени;

- разработан алгоритм проверки свойств трасс выполнения спецификаций в виде формул логики FOTL без вхождения внешних функций и в случае кусочно-постоянных внешних функций;
- разработана и реализована модельная реализация интерпретатора спецификаций на языке ASM с временем, включающая подсистему проверки свойств трасс их выполнения в логике FOTL, а также графический интерфейс пользователя на платформонезависимом языке Java.

Научная новизна. Все полученные в данной работе результаты являются новыми. Впервые предложено детализированное описание временной модели для ASM. Также было разработано подходящее для этой модели расширение языка ASM непрерывным временем. Расширение языка специальными конструкциями для работы с временем позволяет существенно сократить размер спецификаций, время их разработки и уменьшить вероятность внесения пользователем ошибки в текст спецификации. В разработанном подходе контроль над изменением времени компьютерной системы осуществляется интерпретатором, что позволяет сохранять монотонность изменения времени в системе и использовать это при проверке FOTL-свойств.

Практическая ценность. Разработанное в данной работе расширение языка ASM временем может быть использовано для разработки многоуровневых формальных спецификаций, проверки их свойств, а также автоматизации тестирования компьютерных систем с временными ограничениями. В качестве таких компьютерных систем могут выступать программные и программно-аппаратные системы с ограничениями на время реакции, например, автоматизированные системы управления технологическими процессами, автоматизированные системы научных исследований, интерактивное программное обеспечение, например, системы контроля за выполнением экспериментов, охранно-пожарные комплексы мониторинга, транспортные системы управления и диспетчеризации, банкоматы, лифты, и другие подобные системы.

Спецификации на расширенном языке ASM могут быть проверены на реализованном интерпретаторе ASM. С помощью разработанной подсистемы проверки свойств могут быть проверены заданные пользователем ограничительные свойства. Разработанные примеры вместе с интерпретатором также можно использовать при обучении методам формальной спецификации.

Апробация работы. Полученные результаты работы докладывались на следующих научных конференциях и семинарах.

1. Международная конференция «Космос, астрономия и программирование» (Лавровские чтения). СПбГУ, Санкт-Петербург, 2008;

2. Конференция «Процессы управления и устойчивость», СПбГУ, Санкт-Петербург, 2008;
3. Конференция «Научное программное обеспечение в образовании и научных исследованиях». СПбГПУ, Санкт-Петербург, 2008;
4. Четырнадцатый международный семинар ASM'07. Гримстад, Норвегия, 2007;
5. Семинар лаборатории LACL (laboratoire d'algorithmique, complexité et logique). Университет Париж 12, Кретей, Франция, 2007;
6. Семинар, посвященный разработке открытых программных средств для работы с ASM (Pisa Workshop on ASM Open Source Tools). Пиза, Италия, 2007;
7. Четвертая международная конференция FORMATS «Формальное моделирование и анализ систем с временем» (Formal Modeling and Analysis of Timed Systems). Париж, Франция, 2006;
8. Четвертый международный семинар MSVVEIS «Моделирование, симуляция, верификация и валидация промышленных информационных систем» (Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems). Пафос, Кипр, 2006;
9. Семинар кафедры информатики математико-механического факультета. СПбГУ, Санкт-Петербург, 2005.

Публикации. Основные результаты диссертации опубликованы в работах [1-10], перечисленных в настоящем автореферате. В работе [1] диссертантом разработана спецификация распределенной системы электронных торгов с использованием методов формальной спецификации и реализован ее программный прототип, соавторами разработана архитектура системы и ее неформальная спецификация. В работе [2] диссертантом предложены способы организации и хранения знаний о программных проектах, соавтору принадлежат методы конвертации спецификаций. Работа [7] опубликована в издании, входящем в перечень ВАК.

Структура и объем диссертации. Диссертация состоит из введения, заключения, пяти глав, пяти приложений и списка литературы. Главы разбиты на разделы и подразделы. Объем основной части работы — 94 страницы. Список литературы состоит из 121 наименования. Полный объем работы — 151 страница.

Содержание работы

Во Введении приведен обзор современного состояния данной предметной области, обоснована актуальность диссертационной работы, сформулированы цели и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

Во введении также приведен обзор известных и наиболее значимых в данной области проектов в виде публикаций и реализованных инструментальных средств для интерпретации, генерации кода, трансляции и верификации спецификаций. Также анализируются их отличительные свойства и особенности.

В первой главе вводятся основные понятия, относящиеся к методу ASM. Сначала описывается исходная концепция ASM Гуревича (GASM), затем методы расширения синтаксиса и семантики языка спецификаций. В целом, спецификацию системы с временными ограничениями можно задать парой $\langle ENV, ASMSPEC \rangle$, где ENV — это спецификация окружения, которая содержит определения функций окружения, значения задержек для различных операций языка, а также функцию, определяющую способ раскрытия недетерминизмов, а $ASMSPEC$ — это спецификация машины абстрактных состояний Гуревича с временем. ASM с временем, в свою очередь, можно задать тройкой $\langle VOC, INIT, PROG \rangle$, где VOC — словарь функциональных символов, $INIT$ — описание начального состояния машины и $PROG$ — ее программа. Словарь ASM состоит из множества сортов, множества функциональных символов и множества предикатных символов. Некоторые из них имеют фиксированную интерпретацию, например: \mathcal{R} — сорт вещественных чисел, \mathcal{Z} — сорт целых чисел, \mathcal{N} — сорт натуральных чисел, $BOOL$ — сорт булевых значений, $\mathcal{T} = \mathcal{R}_+$ — выделенный сорт значений времени, $UNDEF = \{undef\}$ — специальный сорт, содержащий одно значение, применяющийся для задания неопределенных значений. Метод ASM позволяет использовать несколько уровней абстракции спецификаций, поэтому на самом высоком уровне абстракции сорт вещественных чисел является математическим множеством вещественных чисел. При переходе на более низкие уровни абстракции базовый сорт может быть уточнен, так на уровне реализации вступают в силу программно-аппаратные ограничения конкретной компьютерной системы. В частности, разработанный интерпретатор в качестве уточнения сорта вещественных чисел использует подмножество рациональных чисел, предоставляемое языком реализации.

Далее описываются дополнительные возможности расширенного языка ASM, ориентированного на спецификацию класса систем с ограничениями по времени. Метод ASM не предполагает фиксированной временной модели, так как является универсальным методом описания алгоритмов, поэтому для

спецификации систем рассматриваемого класса вводится временная модель, адаптирующая универсальный метод ASM. В числе дополнений необходимо отметить специальное представление *внешних функций*, которые задают входные данные и формируют окружение. Определение внешних функций формулируется следующим образом. Рассмотрим внешнюю функцию $f : \mathcal{X} \rightarrow \mathcal{Y}$ и обозначим соответствующую ей функцию с временным параметром $f^\circ : \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{Y}$, где f — имя функции, \mathcal{X} — абстрактный сорт или прямое произведение двух абстрактных сортов, \mathcal{T} — временной сорт, \mathcal{Y} — абстрактный сорт или сорт \mathcal{R} . Вариант функции с временным параметром f° непосредственно в программе ASM не используется, чтобы не усложнять семантику языка. Такие функции предназначены только для описания ограничений пользователя. Для определения внешних функций применяется следующий подход. Рассматривается некоторая интерпретация абстрактного конечного сорта \mathcal{X} и множество элементов этой интерпретации нумеруется натуральными числами. Тогда для каждого элемента интерпретации значения некоторой функции f можно задать с помощью индексированной последовательности: $f(i) := (t_1^i, f_1^i; t_2^i, f_2^i; \dots; t_k^i, f_k^i; \dots)$, где $i \in 1, \dots, n$; n — мощность сорта \mathcal{X} ; $t_1^i, t_2^i, \dots, t_k^i, \dots$ — начальные точки интервалов времени; $f_1^i, f_2^i, \dots, f_k^i, \dots$ — значения функции, определенные на интервалах. Начальные точки временных интервалов t_k^i пронумерованы в порядке возрастания. Также поддерживается спецификация функций от двух переменных, т. е. функций, заданных на прямом произведении двух абстрактных сортов $\mathcal{X}' \times \mathcal{X}''$. Тогда определение таких функций принимает вид: $f(i, j) := (t_1^{i,j}, f_1^{i,j}; t_2^{i,j}, f_2^{i,j}; \dots; t_k^{i,j}, f_k^{i,j}; \dots)$, где $i \in 1, \dots, n, j \in 1, \dots, m$; n — мощность сорта \mathcal{X}' ; m — мощность сорта \mathcal{X}'' ; остальные обозначения аналогичны использованным в описании одноместной функции.

Далее в этой главе описываются методы приписывания временных задержек операциям языка спецификации с временем. Для обозначения задержек введем функцию δ . Будем считать, что функция задержки задана на множестве правильных предложений языка спецификации и действует в множество временных значений \mathcal{T} . Пусть \mathcal{S} — множество всевозможных предложений языка. Тогда тип функции задержки будет выглядеть следующим образом: $\delta : \mathcal{S} \rightarrow \mathcal{T}$. Для каждой базовой операции или конструкции значение функции δ явно задается пользователем и используется интерпретатором для расчета задержек более сложных предложений языка. Чтобы получить задержку суперпозиции синтаксических конструкций P_1 и P_2 , например, последовательного блока $\{P_1 P_2\}$, необходимо просуммировать значения задержек всех его составляющих замещений: $\delta(\{P_1 P_2\}) = \delta(P_1) + \delta(P_2)$. Исключением является случай параллельного замещения, к примеру, $[P_1 P_2]$. В данном случае задержка вычисляется как максимальная задержка среди замещений в параллельном блоке, т. е. $\delta([P_1 P_2]) = \max\{\delta(P_1), \delta(P_2)\}$.

Еще одной важной особенностью метода ASM является возможность задания недетерминированных переходов из одного состояния в другое. В данной работе встречается два вида недетерминизма. В одном случае пользователь явно задает недетерминированный выбор элемента из некоторого множества возможных значений с помощью конструкции **choose**. Во втором случае имеет место неявный недетерминизм, возникающий при одновременной коррекции интерпретации функции в одной и той же точке в параллельных ветвях вычисления. В качестве разрешающей процедуры предлагается выбор одного из вариантов: по индексу или по значению, с помощью нескольких функций, либо при помощи генератора случайного индекса элемента. При определенной настройке интерпретатора возможен останов выполнения спецификации.

Во второй главе описываются основные синтаксические особенности языка ASM с временем, а также семантика расширенного языка ASM. В этой главе приводятся и поясняются основные конструкции языка, которые далее встречаются в пояснительных примерах. Подробная грамматика языка ASM с временем представлена в приложении А.

Семантика языка ASM с временем значительно отличается от семантик других известных языков, основанных на методе ASM. Синтаксически конструкции языка выглядят привычным образом, но операции языка имеют ряд дополнительных побочных эффектов. Так, все арифметические действия и операции чтения/записи имеют побочный эффект: изменение состояния глобального счетчика времени. Следовательно, при переходе к следующему состоянию глобальной ASM могут измениться значения внешних функций, что в свою очередь может повлечь за собой изменение потока управления. Модифицирована также семантика условных операторов цикла, которые при определенных условиях выполнения должны «предвидеть» будущие изменения интерпретаций внешних функций.

В третьей главе описывается метод проверки свойств спецификаций ASM, представленных в виде формул языка некоторой логики. В данной работе была выбрана подходящая для этих целей логика первого порядка с временем FOTL [7]. Приводится описание логики FOTL, описываются ее синтаксические и семантические особенности. Словарь FOTL состоит из конечного числа сортов, функциональных и предикатных символов. Каждая переменная привязана к некоторому сорту. Некоторые из сортов имеют фиксированную интерпретацию. Среди них вещественные числа \mathcal{R} и временной сорт \mathcal{T} . Для логики FOTL понятия интерпретации, модели, выполнимости и истинности имеют те же значения, что и для логики предикатов первого порядка. Исключение составляют несколько символов словаря, имеющих фиксированную интерпретацию.

Во втором разделе третьей главы описывается предлагаемый в данной работе метод проверки пользовательских свойств, который можно разделить

на несколько этапов:

- синтаксический анализ формулы;
- элиминация кванторов над абстрактными переменными;
- элиминация функциональных символов;
- элиминация временных переменных;
- получение результата проверки в виде истинностного значения.

В четвертой главе приведено описание программной реализации разработанной программной системы. Описываются архитектура ядра системы и графический интерфейс пользователя. Ядро системы включает в себя синтаксические анализаторы языка ASM с временем и языка FOTL, загрузчик конфигурации, интерпретатор спецификаций, хранилище трасс прогона, а также подсистему проверки свойств спецификаций в виде формул языка FOTL.

В Заключении приведен список основных результатов, полученных в работе. **Приложение А** содержит грамматику языка спецификации ASM с временем. В **Приложении Б** описана грамматика языка задания свойств на основе логики FOTL. **Приложения В и Г** содержат примеры спецификаций с применением разработанного языка ASM с временем. В **Приложении Д** приводится руководство пользователя программной реализации разработанной системы.

Цитированная литература

- [1] *Агафонов, В. Н.* Спецификация программ: понятийные средства и их организация / В. Н. Агафонов. — Новосибирск: Наука, 1990. — 224 с.
- [2] *Баранов, С. Н.* Индустриальная технология автоматизации тестирования мобильных устройств на основе верифицированных поведенческих моделей проектных спецификаций требований / С. Н. Баранов, В. П. Котляров, А. А. Летичевский // *Материалы международной научной конференции «Космос, астрономия и программирование» (Лавровские чтения), СПбГУ, Санкт-Петербург.* — 2008. — С. 134–145.
- [3] *Соловьев, И. П.* Формальные спецификации вычислительных систем. Машины абстрактных состояний (машины Гуревича) / И. П. Соловьев. — Издательство СПбГУ, 1998. — 32 с.
- [4] *Anlauff, M.* XASM — an extensible, component-based ASM language / M. Anlauff // *Abstract State Machines, Theory and Applications, International Workshop, ASM 2000, Monte Verità, Switzerland, March 19-24, 2000* / Ed. by Y. Gurevich, P. W. Kutter, M. Odersky, L. Thiele. — Vol. 1912 of *Lecture Notes in Computer Science.* — Springer, 2000. — Pp. 69–90.
- [5] *AsmL: The Abstract State Machine Language, Foundations of Software Engineering* — Microsoft Research, 2002. — October. <http://research.microsoft.com/fse/asml/>.
- [6] *Beauquier, D.* Decidable verification for reducible timed automata specified in a first order logic with time / D. Beauquier, A. Slissenko // *Theor. Comput. Sci.* — 2002. — Vol. 275, no. 1-2. — Pp. 347–388.
- [7] *Beauquier, D.* A first order logic for specification of timed algorithms: Basic properties and a decidable class / D. Beauquier, A. Slissenko // *Annals of Pure and Applied Logic.* — 2002. — Vol. 113, no. 1–3. — Pp. 13–52.
- [8] *Börger, E.* Logic programming: The evolving algebra approach. / E. Börger // *IFIP 13th World Computer Congress* / Ed. by B. Pehrson, I. Simon. — Vol. I: Technology/Foundations. — Elsevier, Amsterdam, the Netherlands: 1994. — Pp. 391–395.
- [9] *Börger, E.* Report on a practical application of ASMs in software design / E. Börger, P. Päppinghaus, J. Schmid // *Abstract State Machines, Theory and Applications, International Workshop, ASM 2000, Monte Verità, Switzerland, March 19-24, 2000, Proceedings* / Ed. by Y. Gurevich,

- P. W. Kutter, M. Odersky, L. Thiele. — Vol. 1912 of *Lecture Notes in Computer Science*. — Springer, 2000. — Pp. 361–366.
- [10] *Börger, E.* Abstract State Machines: A Method for High-Level System Design and Analysis / E. Börger, R. Stärk. — Springer-Verlag, Berlin, 2003. — 440 pp.
- [11] *Castillo, G. D.* Towards comprehensive tool support for abstract state machines: The ASM workbench tool environment and architecture / G. D. Castillo // Applied Formal Methods — FM-Trends’98 / Ed. by D. Hutter, W. Stephan, P. Traverso, M. Ullmann. — Vol. 1641 of *Lecture Notes in Computer Science*. — Springer-Verlag, 1998. — Pp. 311–325.
- [12] *Cohen, J.* On verification of refinements of timed distributed algorithms / J. Cohen, A. Slissenko // Proc. of the Intern. Workshop on Abstract State Machines (ASM’2000), March 20–24, 2000, Switzerland, Monte Verita, Ticino. Lect. Notes in Comput. Sci., vol. 1912 / Ed. by Y. Gurevich, P. Kutter, M. Odersky, L. Thiele. — Springer-Verlag, 2000. — Pp. 34–49.
- [13] *Dijkstra, E. W.* The structure of the “THE”-multiprogramming system / E. W. Dijkstra // *Communications of the ACM*. — 1968, May. — Vol. 11, no. 5. — Pp. 341–346.
- [14] *Emerson, E. A.* Temporal and modal logic / E. A. Emerson // Handbook of Theoretical Computer Science / Ed. by J. van Leeuwen. — New York, N.Y.: Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990. — Vol. B: Formal Models and Semantics. — Pp. 996–1072.
- [15] *Farahbod, R.* CoreASM: An extensible ASM execution engine / R. Farahbod, V. Gervasi, U. Glässer // Abstract State Machines. — 2005. — Pp. 153–166.
- [16] *Graf, S.* Time in state machines / S. Graf, A. Prinz // *Fundam. Inform.* — 2007. — Vol. 77, no. 1-2. — Pp. 143–174.
- [17] *Gurevich, Y.* Evolving algebras 1993: Lipari Guide / Y. Gurevich // Specification and Validation Methods / Ed. by B. Egon. — Oxford University Press, 1995. — Pp. 9–36.
- [18] *Gurevich, Y.* Sequential abstract-state machines capture sequential algorithms / Y. Gurevich // *ACM Transactions on Computational Logic*. — 2000. — July. — Vol. 1, no. 1. — Pp. 77–111.
- [19] *Gurevich, Y.* The railroad crossing problem: An experiment with instantaneous actions and immediate reactions / Y. Gurevich, J. Huggins // Proceed-

- ings of CSL'95 (Computer Science Logic). — Vol. 1092 of *LNCS*. — Springer, 1996. — Pp. 266–290.
- [20] *Guttag, J. V.* Abstract data types and software validation / J. V. Guttag, E. Horowitz, D. R. Musser // *CACM*. — 1978. — Dec. — Vol. 21, no. 12. — Pp. 1048–1064.
- [21] *Ouimet, M.* Timed abstract state machines: An executable specification language for reactive real-time systems: Tech. rep. / M. Ouimet, M. Nolin, K. Lundqvist: Massachusetts Institute of Technology, Cambridge, 2006.
- [22] *Schmid, J.* Introduction to AsmGofer: Tech. rep. / J. Schmid: Siemens Corporation, 2001.
- [23] *Soloviev, I.* The language of interpreter of distributed abstract state machines / I. Soloviev, A. Usov // *Tools for Mathematical Modeling. Mathematical Research*. — 2003. — Vol. 10. — Pp. 161–170.
- [24] Univ. of Michigan, ASM homepage. <http://www.eecs.umich.edu/gasm/>.
- [25] Using abstract state machines at Microsoft: A case study / M. Barnett, E. Börger, Y. Gurevich et al. // Abstract State Machines, Theory and Applications, International Workshop, ASM 2000, Monte Verità, Switzerland, March 19-24, 2000, Proceedings / Ed. by Y. Gurevich, P. W. Kutter, M. Odersky, L. Thiele. — Vol. 1912 of *Lecture Notes in Computer Science*. — Springer, 2000. — Pp. 367–379.
- [26] *Wirth, N.* Program development by stepwise refinement / N. Wirth // *Communications of the Association of Computing Machinery*. — 1971. — April. — Vol. 14, no. 4. — Pp. 221–227.

Работы автора по теме диссертации по годам

1. П. Васильев, И. Соловьев, А. Усов. Разработка распределенной системы электронных торгов с использованием методов формальной спецификации вычислительных систем // Материалы межвузовской конференции в области современных технологий программирования компании Microsoft, 2-4 декабря 2002, С. 23.
2. П. Васильев, И. Соловьев. Применение инженерии знаний в спецификации программных проектов // Журнал «Компьютерные инструменты в образовании», С.-Петербург, №6, 2003, С. 25–34.

3. П. Васильев. Разработка и верификация спецификаций программных продуктов, основанная на знаниях предметной области // Материалы Санкт-Петербургского конкурса-конференции для студентов, аспирантов и молодых ученых. Санкт-Петербург, декабрь 2003, С. 20.
4. P. Vasilyev. Simulator for Real-Time Abstract State Machines // Proceedings of the 4th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, In conjunction with ICEIS 2006, Paphos, Cyprus, May 2006, pp. 202–205.
5. P. Vasilyev. Simulator for Real-Time Abstract State Machines // Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems, Paris, France, September 25-27, 2006, pp. 337–351.
6. P. Vasilyev. Simulator-Model Checker for Reactive Real-Time Abstract State Machines // Proceedings of the ASM'07 the 14th International ASM Workshop (<http://ikt.hia.no/asm07/>), Grimstad, Norway, 2007.
7. П. Васильев. Расширение языка машин абстрактных состояний Гуревича рациональным временем // Вестник С.-Петерб. ун-та. Сер. 10. 2007. Вып. 4. С. 128–140.
8. П. Васильев. Применение расширенного языка абстрактных машин Гуревича с временем для спецификации протокола IEEE 1394 Root Contention // Материалы конференции «Научное программное обеспечение в образовании и научных исследованиях», СПбГПУ, Санкт-Петербург, 2008. С. 15–21.
9. П. Васильев. Верификация FOTL-свойств машин абстрактных состояний Гуревича с временем // Материалы конференции «Процессы управления и устойчивость», СПбГУ, Санкт-Петербург, 2008. С. 302–307.
10. П. Васильев. Задача спецификации протокола выбора корневого устройства стандарта IEEE 1394 на языке абстрактных машин Гуревича с временем // Материалы международной научной конференции «Космос, астрономия и программирование» (Лавровские чтения), СПбГУ, Санкт-Петербург, 2008. С. 32–43.