ФГБОУ ВО «Московский государственный университет
имени М. В. Ломоносова»

На правах рукописи

Кривчиков Максим Александрович

Формальные модели и верификация свойств программ с использованием промежуточного представления

Специальность 05.13.17 —
«Теоретические основы информатики»

Автореферат

диссертации на соискание учёной степени
кандидата физико-математических наук

Москва — 2015

Работа выполнена на кафедре вычислительной математики механико-математического факультета ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова».

Научный руководитель: **Васенин Валерий Александрович,**
доктор физико-математических наук, профессор

Официальные оппоненты: **Махортов Сергей Дмитриевич,**
доктор физико-математических наук, доцент,
ФГБОУ ВПО «Воронежский государственный университет»,
заведующий кафедрой математического обеспечения ЭВМ

Пальчунов Дмитрий Евгеньевич,
доктор физико-математических наук, доцент,
ФГБУН «Институт математики им. С.Л. Соболева Сибирского отделения Российской Академии Наук»,
ведущий научный сотрудник лаборатории теории вычислимости и прикладной логики

Ведущая организация: ФГБУН «Институт системного программирования Российской Академии Наук»

Защита состоится «21» октября 2015 г. в 16 час. 45 мин. на заседании диссертационного совета Д 501.002.16 на базе ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» по адресу: 119991, Москва, ГСП-1, Ленинские горы, д.1, ФГБОУ ВО «МГУ имени М .В. Ломоносова», механико-математический факультет, аудитория 14-08.

С диссертацией можно ознакомиться в Фундаментальной библиотеке ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (Москва, Ломоносовский проспект, д.27, сектор А, 8 этаж), а также на сайте ИСТИНА (http://istina.msu.ru/dissertation_councils/councils/1230240/).

Автореферат разослан «____» _____ 2015 г.

Учёный секретарь диссертационного совета
Д 501.002.16 на базе
ФГБОУ ВО «МГУ имени М. .В Ломоносова»
доктор физико-математических наук,
профессор Корнев Андрей Алексеевич

Общая характеристика работы

Актуальность темы. Программное обеспечение в настоящее время используется практически во всех сферах деятельности человека, в том числе — в организации быта и при автоматизации производственных процессов, в системах телекоммуникаций и на объектах транспорта, в медицинском оборудовании, в научных исследованиях и при сопровождении объектов критически важных инфраструктур[1], включая оборонный комплекс. Согласно различным источникам, ежегодные потери мировой экономики от просчётов, дефектов и ошибок в программах на 2013 год оцениваются в 312 млрд долларов США[2], приводятся также данные в 59.5 млрд долларов на 2002 год только для экономики США[3]. Известны случаи, когда ошибки в программном обеспечении становились причиной многочисленных травм и даже гибели людей[4]. Такие просчёты, различного рода дефекты и ошибки в исходном коде могут появиться на разных стадиях жизненного цикла программного продукта — от проектирования до модификации на этапе полномасштабной эксплуатации. Это особенно характерно для современных, сложно организованных (логически и архитектурно), больших по объёму программных комплексов[5].

[1]Критически важные объекты и кибертерроризм. Часть 1. Системный подход к организации противодействия / О. О. Андреев [и др.] ; под ред. В. А. Васенин. М. : МЦНМО, 2008; Критически важные объекты и кибертерроризм. Часть 2. Аспекты программной реализации средств противодействия / О. О. Андреев [и др.] ; под ред. В. А. Васенин. М. : МЦНМО, 2008.

[2]*Britton T., Jeng L., Carver G.* Reversible Debugging Software: тех. отч. / University of Cambridge, Judge Business School. Cambdidge, UK, 2013.

[3]*RTI* The economic impacts of inadequate infrastructure for software testing: Planning Report / National Institute of Standards ; Technology. 2002. 02-3 (7007.011).

[4]*Weaver R. A.* The Safety of Software – Constructing and Assuring Arguments: PhD / Weaver Robert Andrew. Department of Computer Science, University of York, 2003; *Borrás C.* Overexposure of radiation therapy patients in Panama: problem recognition and follow-up measures // Revista Panamericana de Salud Pública. 2006. Сент. Т. 20, 2-3. С. 173—187; *Leveson N. G., Turner C. S.* An Investigation of the Therac-25 Accidents // Computer. 1993. Июль. Т. 26, № 7. С. 18—41.

[5]Интеллектуальная система тематического исследования научно-технической информации (ИСТИНА) / В. А. Васенин [и др.] // Обозрение прикладной и промышленной математики. 2012. Т. 19, № 2. С. 239—240.

С позиций минимизации числа дефектов и ошибок в программном обеспечении, особенно на начальных этапах его жизненного цикла, основным технологическим процессом является процесс верификации, который и будет рассматриваться далее. В инженерии программ могут быть выделены следующие четыре основных подхода к верификации[6].

Визуальный контроль (рецензирование, code review) — верификация проводится путём просмотра и анализа исходного текста программы на соответствие предъявляемым к ней требованиям.

Тестирование — верификация проводится путём выполнения программы на заранее заданных наборах входных данных и последующего сравнения полученных результатов с эталонными, полученными на основе требований или с использованием модели или прототипа программного комплекса.

Статический анализ — верификация проводится с помощью автоматического анализа исходного кода путём построения модели программы и сравнения её с моделью, построенной на основе спецификаций.

Формальная верификация — верификация заключается в построении доказательства выполнения требуемых свойств (формальной спецификации) для модели программы (формальной семантики).

При этом из перечисленных подходов только формальная верификация позволяет получить в качестве результата строгое математическое доказательство корректности программы по отношению к предъявляемым к ней требованиям.

Недостатки существующих широко используемых подходов к верификации находят своё отражение в статистике плотности дефектов и ошибок в исходом коде программ[7]. Согласно этой статистике, в 2012 году для 118 проанализированных проектов (среднего и крупного размера) с открытым исходным кодом средняя плотность обнаруженных средством статического анализа дефектов составляла 0.69 на тысячу строк исходного кода.

[6] *Липаев В.* Программная инженерия. Методологические основы. М. : ТЕИС, 2006. Лекция 13.

[7] *Coverity* Coverity Scan : 2012 Open Source Report: тех. отч. / Coverity Inc. 2012.

Авторами указанного исследования это оценивается как хороший результат. Однако, согласно В.В. Липаеву[8], достижима на практике плотность ошибок менее 0.20, а уровень порядка 0.05 ошибок на тысячу строк исходного кода считается близким к минимальному достижимому существующими средствами.

Для проведения формальной верификации, как правило, используется формальная модель (семантика) языков программирования, применяемых при разработке программы. В этой связи следует отметить тот факт, что в настоящее время немалая часть программных продуктов разрабатывается с использованием нескольких языков программирования одновременно. Таким образом, на практике для верификации может появиться необходимость рассматривать формальную семантику одновременно нескольких языков программирования. Как следствие, встаёт вопрос о формальной модели, допускающей такое представление.

С учётом изложенного выше, актуальность направления исследований диссертационной работы автора определяется неудовлетворённой на настоящее время практической потребностью в методах и средствах формальной верификации кода крупных, сложных по своей структуре программных комплексов, при разработке которых используется несколько языков программирования.

Объектом диссертационного **исследования** являются математические модели и основанные на них средства разработки и анализа исходного кода программ с целью реализации требований по их формальной верификации. Такие модели и средства должны допускать использование нескольких языков программирования при разработке программ, подлежащих верификации.

В качестве **предмета исследования** рассматриваются процессы описания формальных моделей программ и языков программирования, используемых в исходных текстах программ.

[8] *Липаев В.* Программная инженерия. Методологические основы. М. : ТЕИС, 2006. Лекция 10.

Целью диссертационного исследования является разработка математических моделей и основанных на них программных средств, которые, в свою очередь, предназначены для построения формальных моделей программ и верификации их функциональных свойств с помощью промежуточного представления таких программ на основе λ-исчисления с зависимыми типами.

Для достижения этой цели были поставлены и решены следующие, перечисленные далее **задачи**.

1. Исследовать существующие и разработать новую разновидность λ-исчисления с зависимыми типами для его использования в качестве промежуточного представления при описании формальной семантики различных языков программирования.
2. На основе новой разновидности λ-исчисления с зависимыми типами разработать макеты языка и программного средства для построения формальных моделей и верификации функциональных свойств программ.
3. Исследовать возможность использования разработанного макета для описания формальной семантики различных востребованных языков программирования.
4. Исследовать возможность использования разработанного макета для описания различных аспектов формальной семантики программ, в частности — механизмов описания параллельных программ.

Научная новизна работы заключается в том, что автором:

1. предложена новая разновидность λ-исчисления с зависимыми типами, обеспечивающая поддержку нетривиальных типов идентичности путём введения дополнительных, не рассматривавшихся ранее правил редукции для элементов типов идентичности;
2. на основе предложенной разновидности исчисления реализованы макеты языка программирования и программного средства, которые могут представлять спецификации, существенно использующие нетривиальные типы идентичности с помощью предложенных автором правил редукции;

3. построена новая модель статической формальной семантики промежуточного кода, соответствующего подмножеству стандарта ECMA-335, поддерживающая обобщённые типы согласно четвёртой редакции стандарта;
4. построена новая формальная модель динамического параллельного исполнения программ, которая рассматривается как модификатор формальной семантики языка программирования.

Теоретическая и практическая значимость. Результаты исследований, представленные в диссертационной работе, призваны способствовать развитию моделей, методов и средств разработки программного обеспечения, позволяющие более эффективно реализовать предъявляемые к нему требования с помощью механизмов формальной верификации. Разработанный автором макет языка и средства верификации допускают более высокоуровневое представление программ, написанных на нескольких языках программирования. Такой язык и средства верификации могут эффективно применяться при верификации существующих программ. Предложенная разновидность λ-исчисления с зависимыми типами и реализация базового языка на её основе может быть использована для исследований в области компьютерной математики, в частности, при развитии гомотопической теории типов[9].

Методика исследования. В работе применяются методы теории категорий, теории доменов, математической логики и теории доказательств, теории графов, а также методы программной инженерии и функционального программирования.

Достоверность полученных результатов и выводов диссертации определяется строгими математическими доказательствами и практическими испытаниями, а также тем фактом, что основные результаты и выводы опубликованы в 11 печатных работах и прошли апробацию в форме докладов на ряде научных и научно-практических конференций и семинаров.

[9] *The Univalent Foundations Program* Homotopy Type Theory: Univalent Foundations of Mathematics. Princeton, NJ : Institute for Advanced Study, 2013.

Апробация работы. Основные результаты диссертации докладывались на следующих конференциях и семинарах:

- международная конференция «Мальцевские чтения — 2014», секция алгебро-логических методов в информационных технологиях (Институт математики им. С.Л. Соболева СО РАН, г. Новосибирск, 10–13 ноября 2014 г.);
- научные конференции «Ломоносовские чтения — 2012, 2013, 2014», секция механики (Московский государственный университет имени М.В. Ломоносова, г. Москва, 16–25 апреля 2012 г., 15–23 апреля 2013 г., 14–23 апреля 2014 г.);
- Четвёртая научно-практическая конференция «Актуальные проблемы системной и программной инженерии (АПСПИ — 2015)» (МИЭМ НИУ ВШЭ, г. Москва, 20–21 мая 2015 г.);
- Третья конференция «Информационная безопасность АСУ ТП КВО» (РАНХ и ГС, г. Москва, 29–30 января 2015 г.);
- семинар «Проблемы современных информационно-вычислительных систем» под руководством д.ф.-м.н., проф. В.А. Васенина (2011, 2013, 2014 г., Механико-математический факультет Московского государственного университета имени М.В. Ломоносова, г. Москва);
- расширенный семинар «Методы суперкомпьютерного моделирования» (Институт космических исследований РАН, г. Таруса, 1–3 октября 2014 г.);
- семинар «Интеллектуальные системы» под руководством к.т.н. Ю.А. Загорулько (2014 г., Институт систем информатики им. А.П. Ершова СО РАН, г. Новосибирск);
- семинар «Теоретические проблемы программирования» под руководством д.ф.-м.н., проф. Р.И. Подловченко и д.ф.-м.н., проф. В.А Захарова (2014 г., Факультет вычислительной математики и кибернетики Московского государственного университета имени М.В. Ломоносова, г. Москва);

- семинар отдела «Технологий программирования» под руководством д.ф.-м.н., проф. А.К. Петренко (2014 г., Институт системного программирования РАН, г. Москва).

Часть работ диссертационного исследования производилась в рамках научно-исследовательской работы №1/404-11 Института механики МГУ имени М.В. Ломоносова «Разработка алгоритма распараллеливания теплогидравлического кода (CMS) полномасштабной модели» по договору с ОАО «ВНИИАЭС».

Личный вклад. В диссертации представлены результаты исследований, выполненные автором самостоятельно. Личный вклад автора состоит: в разработке теоретических методов решения задач исследования; в доказательстве утверждений, которые позволяют подтвердить достоверность полученных результатов; в их обработке, анализе и обобщении; в формулировке выводов и направлений дальнейших исследований.

Публикации. Основные результаты по теме диссертации изложены в 11 печатных работах, 5 из которых [1–5] изданы в периодических изданиях, входящих в Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание учёной степени кандидата наук, на соискание учёной степени доктора наук, а 2 [1, 3] — переведены и опубликованы в изданиях, индексируемых Web of Science и Scopus. Список работ приведён в заключительной части настоящего автореферата [1–11].

Структура и объем диссертации. Работа состоит из введения, четырёх глав, заключения, списка литературы. Объем диссертации — 153 страницы, приложений — 39 страниц. Список литературы включает 191 работу.

Содержание работы

Во **Введении** обоснована актуальность исследований, которые проводились в рамках данной диссертационной работы, определены цель и задачи работы, сформулированы научная новизна и практическая значимость работы.

В **первой главе** представлен библиографический обзор и критический анализ публикаций, отражающих предысторию (раздел 1.1) и современное состояние исследований (раздел 1.2) в области разработки формальных моделей программ и языков программирования, а также в области моделей вычислений, основанных на λ-исчислении. По итогам обзора в разделе 1.3 представлена постановка четырёх задач исследования. Решению этих задач посвящены следующие далее главы диссертации.

В рамках исторической библиографии, основными отражёнными в ней аспектами являются: исторические модели вычислений; разделение методов формальной верификации на два основных направления — теоретико-модельные (model checking) и теоретико-доказательные (дедуктивная верификация); появление трёх подходов к описанию формальной семантики языков программирования — аксиоматической, операционной и денотационной семантики. Теоретико-модельные методы используют описание программы в терминах некоторой модели, допускающей исчерпывающую проверку состояний на предмет выполнения программой требуемых свойств. Для теоретико-доказательных, или дедуктивных методов доказательство выполнения свойств получают как вывод в некоторой формальной системе, в которой задана формальная семантик программы. На настоящее время известны результаты успешного применения к задачам формальной верификации методов каждого из классов. Тем не менее, наиболее сложным на этом направлении остаётся вопрос о представлении программ и их спецификаций в виде, допускающем эффективное использование тех или иных методов формальной верификации.

Основные подходы к описанию формальной семантики языков программирования сформировались к началу 1970-х годов, однако применение таких подходов к существующим распространённым языкам программирования на настоящее время остаётся сложной задачей. Наличие формальной семантики не требуется, в частности, международными комитетами по стандартизации (такими как ISO/IEC JTC1 или ECMA) при утверждении стандарта языка, а исследования в области построения формальной

семантики существующих языков носят, в первую очередь, инициативный характер.

В рамках анализа современного состояния исследований отмечены важные результаты по приложению дедуктивных методов формальной верификации, в особенности — с использованием языков программирования с зависимыми типами. В число таких результатов входят компилятор языка C CompCert[10] и микроядро SeL4[11]. Указывается также понятие фундаментального кода, несущего доказательства[12] — способа распространения программы, при котором совместно с исполнимым кодом программы распространяется сертификат, содержащий в машиночитаемом виде доказательства выполнения свойств программой. Отмечается также применимость для этого подхода моделей на основе λ-исчисления с зависимыми типами.

Во **второй главе** представлено решение первой задачи — исследование существующих и разработка новой разновидности λ-исчисления с зависимыми типами для его использования в качестве промежуточного представления при описании формальной семантики различных языков программирования. Раздел 2.1 имеет вводный характер и содержит определение одной из известных разновидностей λ-исчисления с зависимыми типами на основе Расширенного исчисления конструкций[13]. Лямбда-исчисление, как формальная система, вводится в виде трёх отношений (типизации, совместимости и редукции) на наборе допустимых формул (термов) с помощью правил вывода. Термы исчисления перечислены в таблице 1.

Типы идентичности позволяют формулировать и применять в доказательствах спецификации, использующие понятие равенства. Классиче-

[10] *Leroy X.* Formal Verification of a Realistic Compiler // Communications of the ACM. 2009. Т. 52, № 7. С. 107—115.

[11] Comprehensive Formal Verification of an OS Microkernel / G. Klein [и др.] // ACM Trans. Comput. Syst. 2014. Февр. Т. 32, № 1. 2:1—2:70.

[12] *Necula G. C.* Proof-carrying code // Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '97. ACM Press, 1997. С. 106—119; *Appel A.* Foundational proof-carrying code // Proceedings 16th Annual IEEE Symposium on Logic in Computer Science. IEEE Comput. Soc, 2001. С. 247—256.

[13] *Luo Z.* ECC, an extended calculus of constructions // [1989] Proceedings. Fourth Annual Symposium on Logic in Computer Science. IEEE Comput. Soc. Press, 1989. С. 386—395.

Таблица 1: Синтаксис Расширенного исчисления конструкций

Исчисление вводится в форме индексов де Брёйна: вместо переменных используются неотрицательные целые числа, указывающие, на сколько уровней выше в представлении терма в виде дерева была определена переменная, которую необходимо подставить. Операция $\downarrow_k$ используется для подстановки терма вместо переменной с индексом k. Правила редукции определяют преобразования термов, с использованием которых моделируется процесс вычислений. Запись $t : T$ в этой таблице и далее означает, что терм t имеет тип T.

Название	Запись	Примечание
Тип	$\mathrm{Type}_i : \mathrm{Type}_j$	универсальное множество, индексы уровней универсумов i,j составляют решётку
Зависимое произведение	$\Pi(x : A).B : \mathrm{Type}$	тип функций, тип результата которых зависит от значения аргумента
λ-абстракция	$\lambda(x : A).b : \Pi A.B$	редукция: $(\lambda(x : A).b) \cdot a \longrightarrow b \downarrow a$
Приложение	$x \cdot a : B \downarrow x$	
Зависимая сумма	$\Sigma(x : A).B : \mathrm{Type}$	тип пар, в которых тип второго элемента зависит от значения первого элемента
Пара	$\mathrm{pair}_{\Sigma A.B}(a, b) : \Sigma A.B$	редукция: $\mathrm{split}(C, \mathrm{pair}(a, b), r) \longrightarrow r \downarrow_1 a \downarrow b$
Разделение пары	$\mathrm{split}(C, x, r) : C \downarrow x$	

Определены в разделе 2.2, не входят в Расширенное исчисление конструкций:

Тип идентичности	$a_1 =_A a_2 : \mathrm{Type}$	
Рефлексивность	$\mathrm{refl}_A a : a =_A a$	редукция: $J(A, a, a, X, \mathrm{refl}_A a, x) \longrightarrow x$
Удаление типа идентичности	$J(A, a, b, C, \gamma, x) : C \downarrow_1 b \downarrow \gamma$	

ское определение типов идентичности в последние годы было расширено в рамках гомотопической теории типов[14]. Однако правило редукции, используемое в классическом определении (указано в последних строках таблицы 1) недостаточно полно описывает ожидаемое поведение таких расширенных типов с позиции вычислений. Предложенные автором новые правила редукции для расширенной интерпретации типов идентичности в исчислении указаны в разделе 2.2 (определения 2.11, 2.14–2.17). Пример такого правила для зависимых произведений (входит в определение 2.15):

J(A, a, b, ΠE.F, γ, x) $\longrightarrow$ **let** (PI-EQ-REDUCTION)

[14] *The Univalent Foundations Program* Homotopy Type Theory: Univalent Foundations of Mathematics. Princeton, NJ : Institute for Advanced Study, 2013.

$$
\begin{array}{lcl}
\mathrm{E}_0,\ \mathrm{E}_1 & : & \mathbf{Type} \\
\mathrm{E}_0 & \equiv & \mathrm{E} \downarrow_1 \mathrm{a} \downarrow \mathbf{refl}\ \mathrm{a} \\
\mathrm{E}_1 & \equiv & \mathrm{E} \downarrow_1 \mathrm{b} \downarrow \gamma \\
\varepsilon & : & \mathrm{E}_1 =_{\mathbf{Type}} \mathrm{E}_0 \\
\varepsilon & \equiv & \mathrm{J}(\mathrm{A},\ \mathrm{a},\ \mathrm{b},\ \mathrm{E} =_{\mathbf{Type}} \mathrm{E}_0,\ \gamma,\ \mathbf{refl}\ \mathrm{E}_0) \\
\mathrm{e} & : & \mathrm{E}_0 \\
\mathrm{e} & \equiv & \lambda \mathrm{E}_1\ .\ \mathrm{J}(\mathbf{Type},\ \mathrm{E}_1,\ \mathrm{E}_0,\ 1,\ \varepsilon,\ 0) \\
\mathrm{e}' & \equiv & \lambda(\mathrm{e} : \mathrm{E}_0).(\mathrm{h} : \mathrm{A}).(\eta : \mathrm{a} =_{\mathrm{A}} \mathrm{h}).\ \mathrm{J}(\mathrm{A},\ \mathrm{a},\ \mathrm{h},\ \mathrm{E},\ \eta,\ \mathrm{e}) \\
\mathrm{F}' & \equiv & (\mathrm{h},\ \eta) \mapsto \mathrm{F} \downarrow_2 \mathrm{h} \downarrow_1 \eta \downarrow (\mathrm{e}' \cdot (\mathrm{e} \cdot 2) \cdot 1 \cdot 0) \\
 & \mathbf{in} & \lambda \mathrm{E}_1\ .\ \mathrm{J}(\mathrm{A},\ \mathrm{a},\ \mathrm{b},\ \mathrm{F}',\ \gamma,\ \mathrm{x} \cdot (\mathrm{e} \cdot 0)\ )\qquad .
\end{array}
$$

Для новых правил автором доказываются свойства, согласно которым такие правила сохраняют типизацию. Указанные свойства сформулированы и доказаны автором в леммах 2.4–2.7, а в теореме 2.1 обосновывается приложение таких правил к термам удаления типов идентичности, шаблоны подстановки которых (терм C в последней строке таблицы 1) не охвачены определениями 2.11, 2.14–2.17 (более строго понятие нейтральной формы вводится в определении 2.19).

Теорема 2.1. К термам удаления типов идентичности в пустом контексте с типами-шаблонами подстановки, которые находятся в нейтральной форме, могут быть применены правила редукции.

В качестве итогового результата сформулирована и доказана теорема 2.2:

Теорема 2.2. В представленной разновидности исчисления возможна редукция любого корректно типизированного терма удаления типа идентичности в другой терм, имеющий тип, совместимый с типом исходного терма.

Понятие совместимости в формулировке теоремы связано со структурой решётки на индексах типов Type_j, которыми обусловлено использование несимметричного термина «совместимость» термов вместо симметричного «равенство».

Таким образом, показано, что основные положения гомотопической теории типов могут быть охвачены с помощью синтаксических правил ре-

дукции. Вопросы нормализации исчисления с такими правилами рассматриваются в качестве одного из основных направлений дальнейших исследований по представленному в главе 2 результату. Однако они выходят за рамки целей настоящего исследования. По нетривиальной структуре новых правил можно сделать вывод о том, что доказательство свойств нормализации для них является сложной задачей.

Известные исследования по разработке новых разновидностей λ-исчисления с зависимыми типами с интерпретацией типов идентичности в терминах гомотопической теории типов используют, как правило, подходы, отличные от введения дополнительных правил редукции. Такие подходы включают описание исчисления с явными контекстами подстановок[15] или интерпретацию термов на уровне модели исчисления[16]. Ни для одного из методов на настоящее время доказательств нормализации получено не было.

Третья глава посвящена решению второй задачи. В этой главе представлены макеты языка программирования и программного средства, которые, в свою очередь, предназначены для построения формальных моделей и верификации функциональных свойств программ. В первую очередь, в разделе 3.1 описанная в предыдущей главе разновидность λ-исчисления с зависимыми типами дополняется термами, с использованием которых могут быть заданы значения. В число таких термов входят стандартные типы конечных множеств (обозначаются как #0 — пустое множество, #1 — множество из одного элемента и т.д.) и натуральных чисел. Кроме того, в соответствии с основными положениями гомотопической теории типов[17], вводится понятие высших индуктивных типов, с использованием которых определяются индуктивные и коиндуктивные типы.

[15] *Licata D. R.*, *Harper R.* Canonicity for 2-dimensional type theory // Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. T. 47. Philadelphia, PA, USA : ACM, 2012. C. 337—348.

[16] *Bezem M.*, *Coquand T.*, *Huber S.* A model of type theory in cubical sets // 19th International Conference on Types for Proofs and Programs (TYPES 2013). T. 26. 2014. C. 107—128; *Atkey R.*, *Ghani N.*, *Johann P.* A relationally parametric model of dependent type theory // ACM SIGPLAN Notices. T. 49. ACM, 2014. C. 503—515.

[17] *The Univalent Foundations Program* Homotopy Type Theory: Univalent Foundations of Mathematics. Princeton, NJ : Institute for Advanced Study, 2013.

Высший индуктивный тип представляет собой тип, для некоторых элементов которого определяются дополнительные связывающие их элементы типов идентичности. Это понятие может быть проиллюстрировано математическим понятием фактор-множества по некоторому классу эквивалентности, в котором такие дополнительные элементы типов идентичности определяются отношением эквивалентности на исходном множестве, или аналогичным понятием фактор-пространства в топологии. Известны и более сложные примеры, близкие к формальным спецификациям программ, например, спецификация репозитория для системы контроля версий данных[18]. В настоящей работе используется описание высших индуктивных типов в терминах фактор-типов. Такие фактор-типы состоят из типа-носителя и семейства типов, для каждой пары элементов носителя задающего дополнительные способы отождествления таких элементов, при наличии таких способов. Для определения используются дополнительные правила вывода и редукции.

Индуктивные типы — это типы, элементы которых задаются по индукции. Более строго это понятие может быть введено в терминах инициальной алгебры функтора (теория категорий) или в терминах наименьших неподвижных точек. Если дана некоторая функция $F : \Pi\text{Type}.\text{Type}$, то такая функция может задавать некоторый индуктивный тип $\mu F : \text{Type}$, который представляет собой наименьшую неподвижную точку последовательности $F^n(\#0)$, где $\#0$ — тип, не имеющий элементов, соответствующий ложному высказыванию. Для работы с такими типами требуются также термы введения ($\mathbf{in} : \Pi(F(\mu F)).\mu F$) и удаления ($\mathbf{fold} : \Pi T.(\Pi F(\mu F).T).\mu F.T$ и более сложные разновидности этого правила) элементов. Однако для некоторых функций F введение такого типа может привести к парадоксам и, как следствие — к противоречивости допускающего их исчисления. С целью сохранения

[18] Homotopical patch theory / C. Angiuli [и др.] // Proceedings of the 19th ACM SIGPLAN international conference on Functional programming. ACM, 2014. С. 243—256.

непротиворечивости, как правило, используются синтаксические ограничения на F[19] или дополнительные нетривиальные условия типизации[20].

При наличии высших индуктивных типов условия, аналогичные представленным в последней работе, могут быть сформулированы в терминах самого исчисления. В настоящей работе для описания индуктивного типа μF используется высший индуктивный тип с носителем $\Sigma(n : \mathbb{N}).F^n(\#0)$ — некоторым конечным приближением $F^\infty(\#0)$, условие наименьшей неподвижной точки на котором задаётся с помощью функции повышения уровня приближения $(\Pi(F^n(\#0)).F^{n+1}(\#0))$. Для валидации с помощью описанного способа определяется стандартный индуктивный тип списков, для которого показывается, что он обладает всеми необходимыми свойствами, а именно: могут быть введены конструкторы пустого списка и списка, состоящего из элемента и другого типа; доступен зависимый терм поэлементного удаления списка.

Аналогичным образом, с помощью конечных приближений с функцией перехода, определяются коиндуктивные типы — дуальные индуктивным с позиций теории категорий (терминальная коалгебра функтора). Носителем в определении высшего индуктивного типа, соответствующего коиндуктивному, является при этом $\Pi(n : \mathbb{N}).F^n(\#1)$, а отношение эквивалентности определяется с позиций терминальности коиндуктивного типа $(\Pi S.F^n(S).F^n(\#1))$. Для демонстрации используется пример типа бесконечных потоков, необходимые свойства которого дополнительно проверяются в среде Coq. Недостающие положения используемой разновидности исчисления при этом определяются в среде Coq аксиоматически.

Раздел 3.2 описывает основные особенности реализации макета языка программирования и программного средства на основе полученной модели исчисления. Для реализации используется язык и среда исполнения

[19] *Paulin-Mohring C.* Inductive Definitions in the System Coq - Rules and Properties // TLCA '93 Proceedings of the International Conference on Typed Lambda Calculi and Applications. Springer-Verlag London, UK, 1993. С. 328—345. (Lecture Notes in Computer Science).

[20] *Abel A.* Towards Generic Programming with Sized Types // Mathematics of Program Construction / под ред. T. Uustalu. Springer Berlin Heidelberg, 2006. С. 10—28. (Lecture Notes in Computer Science ; 4014).

ECMAScript. Синтаксис языка аналогичен синтаксису языка LISP, с дополнительными видами форм для описания отношения типизации. На этапе преобразования в термы имена переменных, полученные по дереву абстрактного синтаксиса, заменяются на индексы де Брёйна. На следующей стадии выполняется проверка отношения типизации по известной схеме[21]. Для проверки допустимости набора индексов уровней универсумов используется алгоритм Тарьяна выделения сильно связанных компонент в ориентированном графе индексов, рёбра которого имеют метки $\leqslant$ и $<$. Недопустимым является набор, содержащий цикл с ребром, которое помечено как $<$. Для удобства использования в качестве отдельной стадии перед стадией типизации реализован частичный вывод типов, построенный на основе функций совместимости. Типизация выполняется на термах после вывода типов, что позволяет исключить ошибки, потенциально вносимые на стадии вывода типов.

Программная реализация макета была протестирована на различных примерах, включая модели, которые рассматриваются в последующих главах работы. Одним из примеров является известный парадокс, возникающий при отсутствии проверки на допустимость набора индексов уровней универсумов[22]. Отмечено, что при отключении проверки индексов уровней универсумов синтаксическая форма, соответствующая парадоксу, проходит типизацию, имеет тип ложного высказывания (#0), и не имеет нормальной формы. При включённой проверке терм не проходит типизацию.

Таким образом, в третьей главе был построен макет языка и программного средства, предназначенных для построения формальных моделей и верификации свойств программ. В качестве промежуточного представления, используемого при верификации программ, написанных на нескольких языках программирования, применяется представление в виде

[21] *Chapman J. M.* Type checking and normalisation: Ph.D. thesis / Chapman James Maitland. Nottingham, UK : University of Nottingham, окт. 2008.

[22] *Hurkens A. J. C.* A simplification of Girard's paradox // Typed Lambda Calculi and Applications. Т. 902 / под ред. G. Goos [и др.]. Berlin, Heidelberg : Springer Berlin Heidelberg, 1995. С. 266—278.

термов λ-исчисления. Таким представлением в реализации макета языка и программного средства оперирует стадия типизации.

В **четвёртой главе** представлены результаты построения статической формальной семантики управляемого кода, соответствующего стандарту ECMA-335[23], которое выполнено с использованием разработанных автором макетов языка и программного средства. Это соответствует третьей задаче исследования. Раздел 4.1 содержит базовые сведения о стандарте, которые необходимы для лучшего понимания материала. Выбор управляемого кода стандарта ECMA-335 обоснован тем фактом, что в этот промежуточный код могут быть скомпилированы программы, написанные на различных широко используемых языках, в том числе — на языках C# и Java. Модель семантики такого кода в терминах макета языка позволяет использовать разработанный макет в рамках настоящей работы в качестве промежуточного представления формальных моделей программ, написанных с использованием нескольких языков программирования, без необходимости описания семантики таких языков по отдельности.

Основные положения модели статической формальной семантики и процесс построения модели рассматриваются в разделе 4.2. В число представленных в этом разделе результатов входят, в первую очередь, окружения статической семантики, которые ссылкам на различные объявления языка ставят в соответствие объекты, описывающие статическую семантику таких объявлений. Для удобства определений синтаксис макета языка расширяется конструкциями для определения алгебраических типов данных (типов, определяемых в виде конечного набора конструкторов), а метод описания индуктивных типов — взаимно индуктивными типами. Такие расширения раскрываются на этапе преобразования синтаксических форм в термы. При этом структура промежуточного представления и алгоритм типизации не изменяются. Пример взаимно-индуктивного определения для области видимости определений и ссылки на пользовательский тип данных, согласно Partition II.7.3 стандарта ECMA-335:

[23] *ECMA International* Standard ECMA-335 - Common Language Infrastructure (CLI). 4-е изд. Geneva, Switzerland, июнь 2006.

$$
\begin{aligned}
\mathrm{FScope} &\equiv \lambda\ (\mathrm{Scope}\ \mathrm{Class}_{\mathrm{ref}} : \mathbf{Type})\ .\ \mathrm{adt} \\
&\qquad \{\ \mathrm{Top} : (\Sigma\ (\mathrm{asm} : \mathrm{Assembly}_{\mathrm{ref}})\ .\ (\mathrm{mod} : \mathrm{Module}_{\mathrm{ref}}))\ \} \\
&\qquad \mid \{\ \mathrm{Nested} : \mathrm{Class}_{\mathrm{ref}}\ \} \\
\mathrm{FClass}_{\mathrm{ref}} &\equiv \lambda\ (\mathrm{Scope}\ \mathrm{Class}_{\mathrm{ref}} : \mathbf{Type}) \\
&\qquad \Sigma\ (\mathrm{scope} : \mathrm{Scope})\ (\mathrm{name} : \mathrm{Ide}_{\mathrm{dot}})\ (\mathrm{nparams} : \mathbb{N}) \\
\mathrm{scr} &\equiv \mathrm{Ind}_{\Sigma \mathbf{Type}.\mathbf{Type}}\ (\lambda\ (\Sigma \mathbf{Type}.\mathbf{Type})\ .\ (\mathbf{split}\ (\Sigma \mathbf{Type}.\mathbf{Type},\ \mathbf{var}\ 0, \\
&\qquad \lambda\ (\mathrm{s}\ \mathrm{c} : \mathbf{Type})\ .\ \mathbf{pair}\ (\mathrm{FScope} \cdot \mathrm{s} \cdot \mathrm{c},\ \mathrm{FClass}_{\mathrm{ref}} \cdot \mathrm{s} \cdot \mathrm{c}))))) \\
\mathrm{Scope} &\equiv \mathbf{fst}\ \mathrm{scr} \\
\mathrm{Class}_{\mathrm{ref}} &\equiv \mathbf{snd}\ \mathrm{scr}\ .
\end{aligned}
$$

Кроме того, при описании семантики используются стандартные абстрактные типы данных — списки и словари.

Раздел 4.3 включает описание динамической семантики минимального подмножества управляемого кода стандарта ECMA-335, необходимого для демонстрации применения построенной модели статической семантики. Такое подмножество позволяет решить задачу описания динамической семантики минимального фрагмента кода (39 строк кода на языке C), входящего в состав программного комплекса математического моделирования теплогидравлических процессов в первом и втором контурах реакторов АЭС[24]. Для описания динамической семантики используется известный подход на основе монад[25], который адаптирован автором для использования в рамках макета языка и программного средства. Монада представляет собой конструктор типа m : ΠType.Type и две операции — **return** : $\Pi(A : \mathrm{Type}).A.m\,A$ и **bind** : $\Pi(AB : \mathrm{Type}).(f : \Pi A.m\,B).(a : m\,A).m\,B$, на композицию которых наложены ограничения, называемые законами монад (monad laws). Такие объекты, как монады и расширяющие это понятие преобразователи монад позволяют описывать единым образом такие аспекты вычислений, как изменяемое состояние, контекст, исключения, а также

[24] *Kroshilin A. E., Maidanik V. N.* A new approach to calculating parameters of thermohydraulic networks for vapor-gas-liquid flows // Thermal Engineering. 2007. Май. Т. 54, № 5. С. 368—374.

[25] *Moggi E.* Notions of computation and monads // Information and Computation. 1991. Июль. Т. 93, № 1. С. 55—92; *Papaspyrou N. S.* A Formal Semantics for the C Programming Language: Doctoral Dissertation / Papaspyrou Nikolaos S. Athenes, Greece : National Technical University of Athenes, 1998.

предоставляют возможность задавать строгую последовательность операций. Описание выполняется в терминах операций **bind** и **return**, а также дополнительных, зависящих от конструктора m операций (например, операций чтения и записи состояния).

Первым шагом при построении модели динамической семантики подмножества управляемого кода стандарта ECMA-335 является представление целевой виртуальной машины в терминах набора преобразователей монад (стека монад), которые описывают отдельные аспекты исполнения. Затем в терминах операций над стеком монад задаётся семантика инструкций кода. В завершение определяется функция динамической семантики, которая ставит объектам статической семантики в соответствие их динамическую семантику. Для описания динамической семантики чисел с плавающей точкой используется разработанная с участием автора модель вычислений с плавающей точкой. Оригинальная модель для среды Coq адаптирована автором диссертации для использования в произвольной разновидности λ-исчисления с зависимыми типами, поддерживающей зависимые суммы, натуральные числа и конечные типы. Макет базового языка удовлетворяет таким требованиям, что позволило использовать эту модель.

Пятая глава посвящена описанию разработанной автором модели динамического параллельного исполнения программ. Модель позволяет описать квазипараллельное (чередующее атомарные операции в произвольном порядке) исполнение программ в форме модификатора семантики языка программирования. Модель построена с использованием макета языка и программного средства, построенных в третьей главе, и подхода на основе монад, адаптированного к ним в четвёртой главе. В разделе 5.1 представлен язык управления потоком исполнения, который используется для описания порядка запуска и использования результатов работы вычислительных потоков в программе. Такой язык содержит следующие основные команды:

- skip передаёт управление далее, не изменяя локальное состояние;
- comp · c выполняет вычисление, задаваемое в терминах семантики исходного языка значением c;

- seq · **pair**(s_1 , s_2) выполняет последовательно команды s_1 и s_2.
- if-then-else · **tuple**(b, s_t , s_f) получает значение условия b и, если условие истинно, выполняет команду s_t, в противном случае — команду s_f.
- while-do · **pair**(b, s) представляет собой цикл с условием выхода b и телом s.
- spawn · **tuple**(t, c, v) создаёт новый поток, выполняющий функцию t с начальным состоянием, задаваемым c, и сохраняет идентификатор потока в переменной v.
- wait · **pair**(v, m) приостанавливает выполнение текущего потока до завершения потока v, после чего производит слияние локальных состояний с помощью функции слияния m.

Раздел 5.2 описывает преобразователь монад возобновлений[26] и другие монады, используемые для описания семантики. Преобразователь монад возобновлений, совместно с монадой булеана (множества всех подмножеств, powerset) позволяет описывать вычисления, результат которых определён неоднозначно, в виде списка всех возможных результатов таких вычислений. Преобразователь монад возобновлений в реактивной форме[27] позволяет разделить модель динамического параллельного исполнения программ на две независимые части — динамическую семантику языка управления потоком исполнения и модель управляющего ядра системы. Последняя модель описывает процессы планирования исполнения параллельных заданий. В разделе 5.3 описывается построение таких моделей.

Раздел 5.4 содержит доказательства свойств модели, которые существенным образом опираются на расширенную интерпретацию типов идентичности согласно второй главе. Такие свойства позволяют утверждать, что, при достаточно сильных ограничениях на запускаемые параллельно функции (в частности, должно быть возможно статически доказать их независи-

[26] *Papaspyrou N. S.* A Resumption Monad Transformer and its Applications in the Semantics of Concurrency: тех. отч. / National Technical University of Athens, School of Electrical ; Computer Engineering, Software Engineering Laboratory. Athenes, Greece, 2001. CSD-SW-TR-2-01.

[27] *Harrison W. L.* The essence of multitasking // Algebraic Methodology and Software Technology. Springer, 2006. С. 158—172.

мость) при любом планировании потоков параллельная программа завершается с одним и тем же результатом. В разделе 5.5 приведён пример расширения модели и описания семантики параллельного исполнения фрагмента кода, рассмотренного в четвёртой главе.

В **Заключении** представлены выводы по итогам диссертационного исследования. К их числу относятся перечисленные далее.

1. Предложена новая разновидность лямбда-исчисления с зависимыми типами, обеспечивающая поддержку нетривиальных типов идентичности путём введения дополнительных, не рассматривавшихся ранее правил редукции для элементов типов идентичности.
2. Разрабатываемая разновидность исчисления реализована в форме макета языка программирования и программного средства, в которых могут быть использованы спецификации, существенно использующие нетривиальные типы идентичности с помощью реализации предложенных автором правил редукции.
3. Построена новая модель статической формальной семантики промежуточного кода стандарта ECMA-335, поддерживающая обобщённые типы согласно четвёртой редакции стандарта.
4. Построена новая формальная модель динамического параллельного исполнения программ, которая рассматривается как модификатор формальной семантики языка программирования.

С учётом выводов, полученных в ходе исследования и анализа мирового научного опыта разработки языков программирования, представленных автором в расширенной библиографии[28], в качестве основного направления дальнейших исследований процессов описания формальных моделей компьютерных программ и языков программирования следует выделить применение полученных математических моделей и средств

[28] *Васенин В. А.*, *Кривчиков М. А.* Формальные модели программ и языков программирования. Часть 1. Библиографический обзор 1930—1989 гг // Программная инженерия. 2015. № 5. С. 10—19; *Васенин В. А.*, *Кривчиков М. А.* Формальные модели программ и языков программирования. Часть 2. Современное состояние исследований // Программная инженерия. 2015. № 6. С. 24—33.

в рамках методологии языково-ориентированного программирования[29]. Концепция промежуточного представления для дедуктивной верификации программ, реализованная автором в настоящей работе в форме макета базового языка и программного средства, соответствует требованиям к средствам верификации программ, написанных на нескольких предметно-ориентированных языках программирования. Таким образом, в первую очередь для развития на этом направлении требуется разработка средств для эффективного построения формальных спецификаций большого количества предметно-ориентированных языков программирования на основе построенных в настоящей работе макетов. Основные положения развития предложенного подхода в отмеченном направлении сформулированы автором в публикации[30].

Список опубликованных работ по теме диссертации

Статьи в периодических научных изданиях, входящих в Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание учёной степени кандидата наук, на соискание учёной степени доктора наук:

1. М.А. Кривчиков, В.А. Васенин. Статическая формальная семантика стандарта ECMA-335 // Программирование. — 2012. — №4. — С. 3–16. Английский перевод: V.A. Vasenin, M.A. Krivchikov. ECMA-335 Static Formal Semantics // Programming and Computer Software. — 2012. — Vol. 38, no. 4. — P. 183–188. doi:10.1134/S0361768812040056 (Автору М.А. Кривчикову принадлежит формальная модель статической семантики управляемого кода стандарта ECMA-335).
2. М.А. Кривчиков, В.А. Васенин, А.Е. Крошилин, В.Е. Крошилин, А.Д. Рагулин, В.А. Роганов. Распараллеливание расчетного кода улучшенной

[29] *Ward M.* Language-oriented programming // Software - Concepts and Tools. 1994. Т. 15, № 4. С. 147—161.

[30] *Васенин В., Кривчиков М.* Языково-ориентированное программирование для формальной верификации программного обеспечения // Четвёртая научно-практическая конференция. МИЭМ НИУ ВШЭ, Москва : Изд-во НИУ ВШЭ, Москва, 2015. С. 30—31.

оценки «БАГИРА» для моделирования трехмерной теплогидродинамики многофазных сред в составе полномасштабной суперкомпьютерной модели «Виртуальная АЭС» // Программная инженерия. — 2012. — № 6. — С. 15–23. (Автору М.А. Кривчикову принадлежат результаты тестирования производительности существующих алгоритмов решения систем линейных уравнений с разреженными матрицами, анализ отдельных фрагментов кода).

3. М.А. Кривчиков, В.А. Васенин. Модель динамического параллельного исполнения программ // Программирование. — 2013. — №3. — С. 45–59. Английский перевод: V.A. Vasenin, M.A. Krivchikov. A Model of Dynamical Concurrent Program Execution // Programming and Computer Software. — 2013. — Vol. 39, no. 1. — P. 1–9. doi:10.1134/S0361768813010076 (Автору М.А. Кривчикову принадлежит формальная модель динамического параллельного исполнения программ, доказательство утверждений о единственности результатов вычислений в последовательном режиме исполнения и в режиме исполнения по требованию, доказательство утверждения о неравенстве между результатами вычислений в последовательном режиме исполнения и режиме исполнения по требованию, реализация модели на языке Haskell).
4. М.А. Кривчиков, В.А. Васенин. Формальные модели программ и языков программирования. Часть 1. Библиографический обзор 1930 – 1989 гг. // Программная инженерия. — 2015. — № 5. — С. 10–19. (Автору М.А. Кривчикову принадлежит библиографический обзор и критический анализ результатов исследований в области методов построения формальных моделей программ и языков программирования, формальной верификации и развития программной инженерии в целом, опубликованных с 1930 по 1989 гг).
5. М.А. Кривчиков, В.А. Васенин. Формальные модели программ и языков программирования. Часть 2. Современное состояние исследований // Программная инженерия. — 2015. — №6. — С. 24–33. (Автору М.А. Кривчикову принадлежит библиографический обзор и крити-

ческий анализ публикаций российских и зарубежных учёных с 1990 по 2014 гг и выводы по современному состоянию исследований в области методов построения формальных моделей программ и языков программирования и методов дедуктивной верификации на основе лямбда-исчисления с зависимыми типами).

Статьи в сборниках и тезисы докладов научных конференций:

6. M.A. Krivchikov, V.A. Vasenin, A.E. Kroshilin, V.E. Kroshilin, V.A. Roganov. Scalable Three-Dimensional Thermal-Hydraulic Best-Estimate Code BAGIRA // Proceedings of 2012 International Congress on Advances in Nuclear Power Plants (ICAPP'12). — Chicago, USA, 2012. — P. 2042–2050. (Автору М.А. Кривчикову принадлежат результаты тестирования производительности существующих алгоритмов решения систем линейных уравнений с разреженными матрицами, анализ отдельных фрагментов кода).
7. М.А. Кривчиков, В.А. Васенин. К формальному описанию программ для распределенной вычислительной среды грид-архитектуры // Ломоносовские чтения. Тезисы докладов научной конференции. Секция механики. 16-25 апреля. — Изд-во Московского университета — 2012. — С. 32–33. (Автору М.А. Кривчикову принадлежит модель динамического параллельного исполнения программ).
8. М.А. Кривчиков, В.А. Васенин, И.С. Астапов, В.А. Роганов, В.Н. Майданик. Распараллеливание теплогидравлического расчетного кода CMS в составе полномасштабной суперкомпьютерной модели «Виртуальная АЭС» // Ломоносовские чтения. Тезисы докладов. Секция механики. 15–23 апреля 2013 г. — Изд-во Московского университета. — 2013. — С.27–28. (Автору М.А. Кривчикову принадлежит статический анализатор подмножества языка Fortran, анализ связей различных программных модулей в составе кода, DSL-модель фрагмента теплогидравлического расчетного кода, анализ исходного кода отдельных программных модулей).
9. М.А. Кривчиков, В.А. Васенин. Предметно-ориентированные языки с заданной формальной семантикой на основе лямбда-исчисления

с зависимыми типами // Международная конференция «Мальцевские чтения». Тезисы докладов. [Электронный ресурс]. — URL: http://www.math.nsc.ru/conference/malmeet/14/Malmeet2014.pdf . 2014. — С.25–25. (Автору М.А. Кривчикову принадлежит разновидность лямбда-исчисления с зависимыми типами с расширенной интерпретацией типов идентичности с использованием новых, предложенных автором, правил редукции и реализация такой разновидности лямбда-исчисления с зависимыми типами).

10. М.А. Кривчиков, В.А. Васенин. Приложение одной разновидности типизированного лямбда-исчисления к построению формальных моделей программ // Ломоносовские чтения. Тезисы докладов. Секция механики. 14–23 апреля 2014 г. — Изд-во Московского университета. — 2014. — С. 39–39. (Автору М.А. Кривчикову принадлежит реализация подхода к описанию денотационной семантике на основе монад на основе разновидности лямбда-исчисления с зависимыми типами и расширение модели динамического параллельного исполнения программ).
11. М.А. Кривчиков, В.А. Васенин. Языково-ориентированное программирование для формальной верификации программного обеспечения // Материалы четвертой научно-практической конференции «Актуальные проблемы системной и программной инженерии». Сборник научных трудов. — Изд-во НИУ ВШЭ. — 2015. — С.30–31. (Автору М.А. Кривчикову принадлежит схема адаптации методов формальной верификации на основе предложенной разновидности лямбда-исчисления с зависимыми типами к верификации программ в рамках методологии языково-ориентированного программирования).