Московский государственный университет имени М.В. Ломоносова

На правах рукописи

Подымов Владислав Васильевич

Быстрые алгоритмы проверки эквивалентности программ в моделях с полугрупповой семантикой

01.01.09 – Дискретная математика и математическая кибернетика

АВТОРЕФЕРАТ

диссертации на соискание ученой степени

кандидата физико-математических наук

Москва – 2015

Работа выполнена на кафедре математической кибернетики факультета вычислительной математики и кибернетики федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный университет имени М.В. Ломоносова».

Научный руководитель: **Захаров Владимир Анатольевич**,
доктор физико-математических наук, профессор кафедры математической кибернетики факультета вычислительной математики и кибернетики федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный университет имени М.В. Ломоносова»

Официальные оппоненты: **Ломазова Ирина Александровна**,
доктор физико-математических наук, профессор департамента программной инженерии факультета компьютерных наук федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский университет «Высшая школа экономики»

Башкин Владимир Анатольевич,
доктор физико-математических наук, доцент кафедры теоретической информатики факультета информатики и вычислительной техники федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Ярославский государственный университет им. П.Г. Демидова»

Ведущая организация: Федеральное государственное бюджетное учреждение науки Институт систем информатики им. А.П. Ершова Сибирского отделения Российской академии наук

Защита состоится 27 марта в 11 час. 00 мин. на заседании диссертационного совета Д 501.001.44 при Московском государственном университете имени М.В. Ломоносова по адресу: 119991, Москва, ГСП-1, Ленинские горы, д. 1, стр. 52, факультет ВМК, аудитория 685.

С диссертацией можно ознакомиться в Научной библиотеке Московского государственного университета имени М.В. Ломоносова по адресу: 119192, г. Москва, Ломоносовский проспект, д. 27 — а также на официальном сайте факультета ВМК МГУ http://www.cmc.msu.ru в разделе «Диссертации».

Автореферат разослан «_____» _____ 2015 г.

Ученый секретарь
диссертационного совета Д 501.001.44,
доктор физико-математических наук, доцент О.В. Шестаков

Общая характеристика работы

Актуальность и степень разработанности темы исследования. Основная проблема, исследованию которой посвящена настоящая диссертация, — проблема эквивалентности программ — может быть сформулирована следующим образом: выяснить, имеют ли заданные программы π_1, π_2 схожее (одинаковое, эквивалентное) поведение. Существует много строгих формулировок проблемы эквивалентности в различных моделях вычислений. Наиболее известна среди таких формулировок проблема функциональной эквивалентности: поведение программы определяется как реализуемая ей функция преобразования входных данных в выходные данные, а эквивалентность программ — как совпадение реализуемых ими функций. Исследование функциональной эквивалентности затрудняется теоремой Райса-Успенского, из которой следует неразрешимость функциональной эквивалентности в любой модели вычислений, в которой можно реализовать все функции, вычислимые по Тьюрингу. При этом сравнение функциональности программ используется во многих задачах программирования, из чего следует важность исследования функциональной эквивалентности программ. Такое исследование активно проводится в рамках теории схем программ.

В теории схем программ описываются и исследуются характерные структурные особенности программ различных языков и парадигм программирования. Проблема эквивалентности для этих моделей формулируется так, чтобы из эквивалентности схем (объектов модели) вытекала эквивалентность программ, описываемых этими схемами. В начале развития теории схем программ было получено много результатов разрешимости и неразрешимости проблемы эквивалентности в разнообразных моделях программ, например: разрешимость эквивалентности схем Ляпунова-Янова, логико-термальной эквивалентности стандартных схем, древесной эквивалентности рекурсивных схем; неразрешимость эквивалентности стандартных схем, рекурсивных схем, и разрешимость для

некоторых их подклассов; разрешимость и неразрешимость эквивалентности дискретных преобразователей, алгебраических и пропозициональных моделей программ в зависимости от выбора семантики модели. С развитием программирования и расширением области использования вычислительной техники для решения вычислительных задач выяснилось, что таких результатов недостаточно: решение многих практических задач, как, например, оптимизация, верификация и реорганизация кода, требовало разработки эффективных, или быстрых решающих алгоритмов. Понятие быстроты алгоритма может пониматься по-разному в зависимости от рассматриваемой задачи и цели её исследования. В настоящей диссертации быстрыми считаются алгоритмы, завершающие свою работу за полиномиальное время.

Описание быстрых алгоритмов проверки эквивалентности в моделях программ затрудняется тем, что проблема эквивалентности схем Ляпунова-Янова с бесконечной сигнатурой (бесконечным числом различных символов, которые разрешено использовать для описания примитивов программ; здесь — операторов и предикатов) co-NP-полна. При этом схемы Ляпунова-Янова считаются одной из самых простых моделей вычислений теории схем программ. Однако вклад в co-NP-трудность вносится только бесконечностью сигнатуры, но не структурными особенностями программ: эквивалентность схем Ляпунова-Янова с конечной сигнатурой полиномиально разрешима. В настоящей диссертации исследуется вклад структуры программ в трудность проблемы эквивалентности, то есть рассматриваются модели с конечной сигнатурой. Таким образом, быстрота разработанных в диссертации алгоритмов определяется как полиномиальность их времени работы относительно размеров программ.

В настоящей диссертации проблема эквивалентности изучается для пропозициональных моделей последовательных и рекурсивных (функциональных) программ. Программа в пропозициональной модели представляет собой размеченную систему переходов (для последовательных программ) или размеченную контекстно-свободную грамматику (для рекурсивных программ), строящуюся

над конечными алфавитами операторов и логических условий. Содержательно, логическое условие может пониматься как набор значений всех предикатов программы. Чтобы определить поведение пропозициональной программы, необходимо придать значение всем используемым в ней операторам и логическим условиям. Для этих целей в пропозициональных моделях вводится понятие интерпретации, совпадающее с понятием детерминированной динамической модели в динамической логике. Интерпретация состоит из (детерминированной динамической) шкалы и оценки логических условий. Шкалой определяются: множество состояний данных, над которыми программа производит вычисление; состояние данных, с которого начинается вычисление; значение каждого оператора — функция преобразования состояний данных. Оценка логических условий сопоставляет каждому состоянию данных истинное в нём логическое условие. В каждой интерпретации определяется результат вычисления пропозициональной программы. Программы объявляются эквивалентными в интерпретации, если результаты их вычислений в этой интерпретации совпадают. В настоящей диссертации исследуется проблема эквивалентности программ на шкалах: программы эквивалентны на шкале, если они эквивалентны в каждой интерпретации, включающей в себя эту шкалу. Если состояния данных шкалы являются моноидом на множестве операторов, то такая шкала называется полугрупповой. Если состояния данных шкалы частично упорядочены относительно действия операторов, то такая шкала называется упорядоченной. Примером упорядоченной полугрупповой шкалы является свободная шкала: состояния данных такой шкалы являются свободным моноидом. Другой пример — это шкала, состояния данных которой являются моноидом, определяемым произвольным набором соотношений коммутативности ($ab = ba$, где a и b — операторы) и подавления ($ab = b$, где a и b — операторы). Такая шкала далее называется шкалой с коммутативностью и подавлением. Содержательно, соотношение $ab = ba$ означает, что порядок выполнения подряд идущих операторов a, b не влияет на результат вычисления, а соотношение $ab = b$ — что опера-

тор a оказывается фиктивным, если сразу после него выполняется оператор b. Такие соотношения можно легко получить, анализируя переменные, используемые операторами реальной программы.

При исследовании модели рекурсивных программ в настоящей диссертации внимание уделяется классам линейных и металинейных унарных рекурсивных программ. Рекурсивная программа называется унарной, если все её функции, кроме функции ветвления $(if\ x\ then\ y\ else\ z)$, зависят ровно от одного аргумента. Унарная рекурсивная программа называется линейной, если каждый функциональный терм, использованный в её описании, содержит не более одного вхождения символов, определяемых программой, и металинейной, если последнее требование распространяется на все термы, кроме терма, с которого начинается вычисление программы. Анализ проблемы эквивалентности оказывается затруднительным даже для таких скромных по выразительным возможностям классов рекурсивных программ.

Цель работы — разработка быстрых алгоритмов решения проблемы эквивалентности в пропозициональных моделях последовательных и рекурсивных программ.

Основные результаты работы следующие.

1. Разработаны полиномиальные алгоритмы проверки эквивалентности пропозициональных последовательных программ на произвольных упорядоченных шкалах с коммутативностью и подавлением.

2. Установлены достаточные условия полиномиальной разрешимости проблемы эквивалентности линейных унарных рекурсивных программ на упорядоченных полугрупповых шкалах; разработаны полиномиальные алгоритмы решения указанной проблемы эквивалентности при выполнении указанных достаточных условий.

3. Разработаны полиномиальные алгоритмы проверки эквивалентности металинейных унарных рекурсивных программ на свободных шкалах.

При получении основных результатов использовались **методы** теории автоматов, алгебры, теории графов, теории алгоритмов, теории порядков.

Научная новизна. Ранее была известна разрешимость проблемы эквивалентности пропозициональных последовательных программ на упорядоченных шкалах с коммутативностью и подавлением. При этом полиномиальные алгоритмы проверки эквивалентности были известны только для некоторых частных случаев таких шкал: шкалах с коммутативностью (но без подавления); упорядоченных шкалах с подавлением (но без коммутативности); упорядоченных шкалах с коммутативностью и подавлением, удовлетворяющих ограничениям, сильно сужающим возможности использования подавления. В диссертации разработаны полиномиальные алгоритмы проверки эквивалентности на упорядоченных шкалах с коммутативностью и подавлением, не предполагающие никаких дополнительных ограничений на шкалы.

Ранее были описаны достаточные условия полиномиальной разрешимости линейных унарных рекурсивных программ на уравновешенных полугрупповых шкалах. В диссертации этот результат обобщён на существенно более широкий класс упорядоченных шкал.

Полиномиальные алгоритмы решения проблемы эквивалентности металинейных унарных рекурсивных программ на свободной шкале получены впервые.

Теоретическая и практическая значимость. Многие задачи исследования вычислительных моделей в том или ином виде используют сравнение функциональности объектов этих моделей. Исследование сложности проверки эквивалентности объектов вычислительных моделей даёт понимание того, как структура этих объектов влияет на их функциональность, и тем самым предоставляет дополнительные возможности исследования этих объектов. Если проблема эквивалентности вычислительных объектов оказывается полиномиально разрешимой, то это, как правило, означает простоту устройства вычислительных объектов. При этом алгоритмы проверки эквивалентности могут быть ис-

пользованы в решении практических задач программирования: оптимизации, реорганизации (рефакторинга), обфускации, верификации программ, задачах поиска вредоносного кода и многих других.

Степень достоверности, апробация, список публикаций. Основные результаты диссертации сформулированы в виде теорем о полиномиальной разрешимости проблемы эквивалентности в рассматриваемых моделях программ. Доказательством теорем является описание решающих алгоритмов с обоснованием корректности и оценкой времени работы этих алгоритмов. Результаты представлены на конференции "Интеллектуальные системы и компьютерные науки" (2011 год), семинаре "Дискретная математика и её приложения" (2012 год), конференции "Проблемы теоретической кибернетики" (2014 год) и опубликованы в работах [1–7], четыре из них — в журналах из перечня ВАК [1–4]. В работе [1] В.В. Подымову принадлежат формулировка и техника обоснования основного результата статьи. В работе [4] В.В. Подымову принадлежат формулировка и обоснование основного результата статьи.

Структура и объём диссертации. Диссертация состоит из восьми глав, включая введение (первая глава) и заключение (восьмая глава), и следующего за заключением списка литературы. Общий объём рукописи составляет 164 страницы и включает 5 рисунков. Список литературы содержит 115 наименований.

Основное содержание работы

В первой главе (введении): обсуждается основная проблема, исследованию которой посвящена диссертация, — проблема эквивалентности программ; обосновывается актуальность этой проблемы; приводится обзор основных моделей и результатов резрешимости и неразрешимости проблемы эквивалентности в теории схем программ; обосновывается актуальность построения быстрых алгоритмов проверки эквивалентности программ; формулируются положения,

выносимые на защиту, и приводится содержательное пояснение понятий, используемых в этих положениях; обсуждается новизна полученных результатов; обозначается общая структура диссертации.

Во второй главе приводятся определения и обозначения, относящиеся к последовательностям, словам, порядкам, автоматам, полугруппам (в том числе конечнопорождённым моноидам и определяющим соотношениям), соотношениям коммутативности и подавления.

Если A — алфавит, то записью A^* будет обозначаться множество всех слов в этом алфавите. Символом λ обозначается пустое слово.

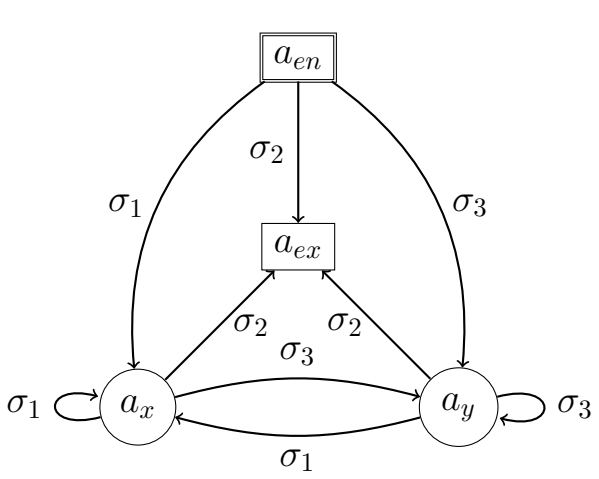
Автомат $\mathfrak{A}$ определяется: входным алфавитом $\mathfrak{A}.A$; конечным множеством состояний $\mathfrak{A}.Q$; множеством меток состояний $\mathfrak{A}.L$; инициальным состоянием $\mathfrak{A}.q \in \mathfrak{A}.Q$; функцией переходов $\mathfrak{A}.T : \mathfrak{A}.Q \times \mathfrak{A}.A \to \mathfrak{A}.Q$; разметкой состояний $\mathfrak{A}.M : \mathfrak{A}.Q \to \mathfrak{A}.L$. На вход автомату $\mathfrak{A}$ подаются слова в алфавите $\mathfrak{A}.A$. Стандартным образом определяется то, как автомат изменяет свои состояния при прочтении символов входного слова. Результатом работы автомата $\mathfrak{A}$ на слове h объявляется метка состояния, в которое автомат переводится этим словом.

Моноид с конечным множеством образующих A, определяемый соотношениями $\mathcal{T}$ вида $h_1 = h_2$, где $h_1, h_2 \in A^*$, может быть описан следующим образом. Элементами моноида являются множества слов в алфавите A. Элемент $\langle h \rangle$, образованный словом h, состоит из всех слов, которые могут быть получены конечным числом применений соотношений множества $\mathcal{T}$ к слову h. Применение соотношения $g_1 = g_2$ состоит в замене в слове h подслова, совпдающего с g_i, на g_{3-i} ($i \in \{1, 2\}$). Соотношение коммутативности имеет вид $ab = ba$, а соотношение подавления — $ab = b$, где $a, b \in A$. Моноид, определяемый произвольным набором соотношений коммутативности и подавления, далее называется моноидом с коммутативностью и подавлением.

В третьей главе приводятся основные определения, относящиеся к рассматриваемым в диссертации моделям программ: пропозициональным моделям последовательных программ (*раздел 3.1*) и рекурсивных программ (*раздел 3.2*).

```
while(x != y) {
  if(x > y) x = x - y;
  else      y = y - x;
}
```

(a) Алгоритм Эвклида



(b) Пропозициональная последовательная программа

Рис. 1. Пример построения пропозициональной последовательной программы

В сигнатуру этих моделей входят конечные алфавиты операторов $\mathfrak{O}$ и логических условий $\mathfrak{L}$.

Пропозициональная последовательная программа π состоит из: конечного множества состояний управления $\pi.L$; входа $\pi.en \in \pi.L$; выхода $\pi.ex \in \pi.L$; функции переходов $\pi.T : L \setminus \{\pi.ex\} \times \mathfrak{L} \to L$; функции привязки $\pi.B : L \to \mathfrak{O}$. Для соотношения $\pi.T(l_1, \sigma) = l_2$ используется запись $l_1 \xrightarrow{\sigma}_\pi l_2$. Объединение отношений $\xrightarrow{\sigma}_\pi$ по всем логическим условиям σ обозначается записью $\to_\pi$. Определение программы поясняется рисунком 1, на котором приведены фрагмент программы в синтаксисе языка Си (рисунок 1(a)) и пропозициональная последовательная программа, описывающая структуру этого фрагмента (рисунок 1(b)). Состояния управления изображены кругами и прямоугольниками, и в них вписаны операторы согласно функции привязки. Двойным прямоугольником изображён вход, одинарным — выход. Операторы a_x, a_y отвечают выполнению инструкций "$x = x - y$" и "$y = y - x$", два других оператора введены для обозначения начала и конца вычисления. Дугами изображена функция переходов программы. Логические условия σ_1, σ_2, σ_3 содержательно описываются соотношениями $x < y$, $x = y$ и $x > y$.

```
Pow :: Integer -> Integer
Pow 0 = 1
Pow x | x>0 = 2*Pow(x-1)
```

(a) Функция вычисления степени двойки

$$D(Pow, \sigma_1) = err$$

$$D(Pow, \sigma_2) = one$$

$$D(Pow, \sigma_3) = dec\ Pow\ dup$$

(b) Описание тел функций унарной рекурсивной программы

Рис. 2. Пример построения унарной рекурсивной программы

В описании унарной рекурсивной программы помимо алфавитов операторов и логических условий рассматривается также счётно-бесконечный алфавит функциональных символов $\mathfrak{F}$. Описание программы и её поведения основывается на понятии терма. Термом называется слово в алфавите $\mathfrak{O} \cup \mathfrak{F}$. Множество термов обозначается записью Terms. Унарная рекурсивная программа π состоит из: конечного множества заголовков $\pi.H \subset \mathfrak{F}$; описания тел функций $\pi.D : \pi.H \times \mathfrak{L} \to \mathsf{Terms}$; запроса $\pi.q \in \mathsf{Terms}$. Введённое определение поясняется рисунком 2, на котором приведены функция вычисления степени числа 2 в синтаксисе языка Haskell (рисунок 2(a)) и описание тел функций соответствующей унарной рекурсивной программы (рисунок 2(b)). Программа содержит единственный заголовок Pow. Логические условия σ_1, σ_2, σ_3 содержательно описываются соотношениями $x < 0$, $x = 0$ и $x > 0$. Операторами one, dec, dup и err описываются соответственно функции 1, $x - 1$, $2x$ и функция, безусловно возвращающая ошибочное значение.

Семантика программ в обеих рассматриваемых моделях определяется на основе шкал и интерпретаций (в терминах динамической логики — (детерминированных) динамических шкал и (детерминированных) динамических моделей соответственно). Шкалой $\mathcal{F}$ определяются: множество состояний данных $\mathcal{F}.S$, над которым программа производит вычисление; начальное состояние данных $\mathcal{F}.s \in \mathcal{F}.S$; функция переходов $\mathcal{F}.R : S \times \mathfrak{O} \to S$. Интерпретация $\mathcal{I}$ включает в себя шкалу $\mathcal{I}.\mathcal{F}$ и разметку состояний данных логическими условиями $\mathcal{I}.\xi : S \to \mathfrak{L}$. Содержательно, шкала придаёт каждому оператору a

значение функции преобразования данных $\overline{a}$: $\overline{a}(s) = \mathcal{F}.R(s,a)$, а интерпретация, помимо этого, придаёт значение логическим условиям. Состояние данных $\overline{a_1}(\overline{a_2}(\ldots\overline{a_k}(\mathcal{F}.s)\ldots))$ обозначаетя записью $[\![a_k\ldots a_1]\!]$. Для состояний данных s_1, s_2 записью $s_1 \prec s_2$ обозначается следующий факт: существует непустая последовательность (или цепочка) операторов $a_1\ldots a_k$, такая что $s_2 = \overline{a_1}(\ldots\overline{a_k}(s_1)\ldots)$. Записью $\preceq$ обозначается рефлексивное замыкание отношения $\prec$. Интерпретация, включающая в себя шкалу $\mathcal{F}$, называется $\mathcal{F}$-интерпретацией. Шкала называется полугрупповой, если существует моноид с множеством образующих $\mathfrak{O}$, такой что: состояния данных шкалы суть элементы моноида; начальное состояние данных есть нейтральный элемент моноида; функция переходов согласована с операцией моноида. Если при этом моноид определяется произвольным набором соотношений коммутативности и подавления, то такая шкала называется шкалой с коммутативностью и подавлением. Шкала называется упорядоченной, если для любых цепочек операторов h, g из равенства $[\![hg]\!] = [\![h]\!]$ следует равенство $g = \lambda$.

Трассой последовательной программы π является всякая последовательность её состояний управления вида $\pi.en \to_\pi l_1 \to_\pi \pi_2 \to_\pi \ldots$. Если трасса программы π оканчивается в выходе, то она объявляется вычислением этой программы. Для простоты вводится обобщение функции привязки $\pi.B$ с состояний данных на трассы: $\pi.B(l_1 \to_\pi l_2 \to_\pi \ldots) = \pi.B(l_1)\,\pi.B(l_2)\ldots$. Трасса $l_1 \to_\pi l_2 \to_\pi \ldots$ реализуется в интерпретации $\mathcal{I}$, если соотношение $l_i \xrightarrow{\mathcal{I}.\xi([\![\pi.B(l_1\to_\pi\ldots\to_\pi l_i)]\!])}_\pi l_{i+1}$ верно для любой пары соседних состояний l_i, l_{i+1} этой трассы. Результатом конечного вычисления cp программы π на шкале $\mathcal{F}$ и в интерпретациях, содержащих эту шкалу, объявляется состояние данных $[\![\pi.B(cp)]\!]$. Бесконечным вычислениям присваивается особый результат $\perp$, не являющийся состоянием данных.

Каждый терм t может быть представлен в виде $t = t.l\ t.\phi\ t.r$, где: $t.l$ — наибольший по длине префикс терма t, являющийся цепочкой операторов; $t.\phi \in \mathfrak{F} \cup \{\lambda\}$. Запись $t \xrightarrow{\sigma}_\pi t'$, где π — унарная рекурсивная программа

и $\sigma \in \mathfrak{L}$, используется для задания терма $t' = t.l\ \pi.D(t.\phi, \sigma)\ \ t.r$. Записью $t \to_\pi t'$ обозначается следующий факт: существует логическое условие σ, такое что $t \xrightarrow{\sigma}_\pi t'$. Трассой программы π является всякая последовательность термов вида $\pi.q \to_\pi t_1 \to_\pi t_2 \to_\pi \ldots$. Вычислением программы π объявляется всякая трасса, которую нельзя продолжить. Трасса $t_1 \to_\pi t_2 \to_\pi \ldots$ реализуется в интерпретации $\mathcal{I}$, если соотношение $t_i \xrightarrow{[\![t_i.l]\!]}_\pi t_{i+1}$ выполнено для любых соседних термов t_i, t_{i+1} этой трассы. Если вычисление оканчивается цепочкой операторов h, то его результатом на шкале $\mathcal{F}$ и в любой $\mathcal{F}$-интерпретации объявляется состояние данных $[\![h]\!]$. Остальные вычисления имеют результат $\perp$, не являющийся состоянием данных.

Программы объявляются эквивалентными в интерпретации, если результаты их вычислений в этой интерпретации совпадают, и эквивалентными на шкале $\mathcal{F}$, или $\mathcal{F}$-эквивалентными, если они эквивалентны в любой интерпретации, включающей в себя эту шкалу. Трассы объявляются совместными на шкале $\mathcal{F}$, или $\mathcal{F}$-совместными, если существует $\mathcal{F}$-интерпретация, в которой реализуются обе эти трассы. Таким образом, программы $\mathcal{F}$-эквивалентны в том и только в том случае, если результаты любых их $\mathcal{F}$-совместных вычислений совпадают.

Исследование модели рекурсивных программ в настоящей диссертации проводится для программ, удовлетворяющим ограничениям линейности и металинейности. Терм называется линейным, если он содержит не более одного вхождения функциональных символов. Унарная рекурсивная программа называется линейной, если все содержащиеся в ней термы линейны, и металинейной, если линейны все термы области значений описания тел функций.

Заголовки унарных рекурсивных программ классифицируются по возможности построения из них конечных вычислений. Например, если программа π', получаемая из π заменой запроса на функциональный символ f, имеет хотя бы одно конечное вычисление, то символ f называется завершаемым в программе π. Также для единообразия завершаемым считается пустое слово.

Для унарных рекурсивных программ вводится понятие нормальной формы, существенно упрощающее анализ этих программ. Коротко оно может быть описано так: во всех термах, содержащихся в программе, разрешается использование только операторов и заголовков программы; термы в описании тел функций обязаны начинаться с оператора; среди заголовков программы ровно один является незавершаемым; запрос состоит только из заголовков программы.

Также в главе 3 сформулированы известные утверждения, не являющиеся заслугой автора настоящей диссертации. Два из них относятся к модели последовательных программ. Первое утверждение формулируется так: для любой трассы программы и любой упорядоченной шкалы $\mathcal{F}$ существует $\mathcal{F}$-интерпретация, в которой реализуется эта трасса. Второе утверждение предоставляет простой способ проверки $\mathcal{F}$-совместности трасс программ для упорядоченной шкалы $\mathcal{F}$, используемый в дальнейших рассуждениях наряду с основным определением совместности. Аналогичные утверждения формулируются для нормализованных унарных рекурсивных программ. Кроме того, для унарных рекурсивных программ приведены утверждения, позволяющие при исследовании проблемы эквивалентности без ограничения общности рассматривать только нормализованные программы.

В четвёртой главе обсуждаются полученные в диссертации результаты (*раздел 4.1*) и подход, с помощью которого эти результаты получены (*раздел 4.2*). Согласно этому подходу — методу совместных вычислений — для проверки $\mathcal{F}$-эквивалентности программ π_1, π_2 вводится и исследуется граф совместных вычислений $\Gamma = \Gamma(\pi_1, \pi_2, \mathcal{F})$, описывающий пары вычислительных конфигураций программ, которые могут быть достигнуты при совместном выполнении этих программ.

Вычислительной конфигурацией рекурсивной программы является терм, а последовательной — состояние данных и состояние управления, достигнутые в процессе вычисления. Совместное выполнение программ в заданной интерпретации начинается во входах (для последовательных программ) или запросах

(для рекурсивных программ). Шаг совместного выполнения состоит следующем. По текущей паре конфигураций определяются активные программы: одна, обе или ни одной. Конфигурации активных программ преобразуются согласно построению вычислений: трасса последовательной программы продолжается на одно состояние управления, а рекурсивной — на один терм. Активность программ задаётся стратегией совместного выполнения. Эта стратегия зависит от выбранной модели программ и от шкалы и для каждой пары конфигураций определяет, какие из программ активны.

Затем каждая пара конфигураций в совместном выполнении заменяется на их различие. Различие конфигураций подбирается таким образом, чтобы по нему можно было определить: завершилась ли одновременная работа программ; если она завершилась, то получены ли вычисления с одинаковыми результатами; какие программы активны согласно выбранной стратегии совместного выполнения; каким будет различие следующей в совместном выполнении пары конфигураций.

Вершинами графа Г объявляются всевозможные различия пар вычислительных конфигураций. Дугами в графе Г соединяются различия, являющиеся соседними хотя бы в одном совместном выполнении программ. Корнем графа Г объявляется различие инициальных конфигураций программ (для последовательной программы — вход и состояние данных, в которое приводит выполнение оператора, приписанного входу; для рекурсивной программы — запрос). В графе Г выделяются опровергающие маршруты — маршруты, отвечающие построению совместных вычислений программ с различными результатми. Таким образом, проверка эквивалентности программ сводится к проверке отсутствия опровергающих маршрутов в графе совместных вычислений. Граф Г, вообще говоря, бесконечен, поэтому из такого сведения не следует разрешимости, а тем более полиномиальной разрешимости $\mathcal{F}$-эквивалентности программ. Алгоритмы проверки $\mathcal{F}$-эквивалентности программ, разработанные в диссертации, за полиномиальное относительно размеров программ время обходят ограничен-

ный фрагмент графа совместных вычислений и по завершении обхода определяют, эквивалентны ли программы. Для каждого из основных результатов диссертации описывается такой обход и обосновываются его корректность и полиномиальная сложность.

В пятой главе обосновывается полиномиальная разрешимость проблемы эквивалентности пропозициональных последовательных программ на произвольных упорядоченных шкалах с коммутативностью и подавлением. Для этого в разделах 5.1–5.3 проводится исследование свойств моноидов с коммутативностью и подавлением, в разделах 5.4–5.6 описывается и исследуется граф совместных вычислений, а в разделе 5.7 — алгоритм проверки эквивалентности, основанный на обходе этого графа. Символом $\mathcal{F}$ далее обозначается произвольная упорядоченная шкала с коммутативностью и подавлением.

В *разделе 5.1* устанавливается критерий упорядоченности моноида с коммутативностью и подавлением.

Теорема 5.1.1. *Пусть* $\mathcal{C}, \mathcal{A} \subseteq \mathfrak{O} \times \mathfrak{O}$ *и* $\mathcal{M}$ *— моноид, образованный множеством* $\mathfrak{O}$ *и определяемый соотношениями* $\{ab = ba \mid (a, b) \in \mathcal{C}\} \cup \{ab = b \mid (a, b) \in \mathcal{A}\}$. *Тогда моноид* $\mathcal{M}$ *упорядочен в том и только в том случае, если* $\mathcal{C}^* \cap \mathcal{A}^+ = \emptyset$, *где* $\mathcal{C}^*$ *— рефлексивно-симметричное замыкание отношения* $\mathcal{C}$, *а* $\mathcal{A}^+$ *— транзитивное замыкание отношения* $\mathcal{A}$.

В *разделе 5.2* исследуются свойства упорядоченных моноидов с коммутативностью и подавлением. С помощью этих свойств обосновываются результаты следующих разделов.

В *разделе 5.3* описан автомат, распознающий подавление. Предполагается заданным упорядоченный моноид $\mathcal{M}$ с коммутативностью и подавлением. Автомат, распознающий подавление, обладает следующими свойствами. На вход автомату подаются слова в алфавите образующих моноида $\mathcal{M}$. Метками автомата являются множества образующих. Результатом работы автомата на слове h является множество образующих a, для которых верно равенство $\langle (ah)^- \rangle = \langle h^- \rangle$ (минусом в верхнем индексе здесь обозначен зеркальный образ слова). Кроме

того, если верно равенство $\langle h^- \rangle = \langle g^- \rangle$, то слова h и g приводят автомат в одно и то же состояние.

Теорема 5.3.1. *Для любого упорядоченного моноида с коммутативностью и подавлением существует автомат, распознающий подавление.*

Автомат, распознающий подавление, используется в разделе 5.5 при построении автомата, распознающего эффекты подавлений.

В *разделе 5.4* приводится описание графа совместных вычислений $\Gamma = \Gamma(\pi_1, \pi_2, \mathcal{F})$, и доказывается сводимость проверки $\mathcal{F}$-эквивалентности к проверке отсутствия опровергающих маршрутов в графе Γ. Структура графа Γ определяется выбором стратегии совместного выполнения и различия конфигураций, приведённых далее. Пусть при совместном выполнении программами π_1, π_2 получены состояния данных s_1, s_2 соответственно. Если $s_1 = s_2$, то активны обе программы. Если $s_2 \prec s_1$, то активна только программа π_2. Иначе активна только программа π_1. При этом если активная программа завершила построение вычисления, то она становится неактивной. В различии пары конфигураций в явном виде сохранены текущие состояния управления программ, а текущие состояния данных s_1, s_2 преобразуются следующим образом: состояния данных s_1, s_2 представляются в виде $s_1 = s \circ s'_1$, $s_2 = s \circ s'_2$; выбирается состояние данных s, такое что кратчайшая приводящая в него цепочка операторов имеет наибольшую возможную длину; в различии сохраняются состояния данных s'_1, s'_2. Корректность такого определения различия основывается на свойствах моноидов с коммутативностью и подавлением, исследованных в разделах 5.1–5.3.

Опровергающие маршруты графа Γ могут быть описаны в терминах совместного выполнения следующим образом. Случай 1: в результате совместного выполнения программами построены конечные вычисления с различными результатами. Случай 2: совместное выполнение бесконечно, программа π_i почти всегда (то есть всегда, кроме быть может, конечного числа шагов) неактивна, при этом её трасса может быть достроена до конечного вычисления ($i \in \{1, 2\}$).

Теорема 5.4.1. *Пусть $\mathcal{F}$ — упорядоченная шкала с коммутативностью и подавлением и π_1, π_2 — пропозициональные последовательные программы. Тогда эти программы $\mathcal{F}$-эквивалентны в том и только в том случае, если граф* Γ *не содержит опровергающих маршрутов.*

В *разделе 5.5* вводится аппарат эффектов подавлений, необходимый для ограничения числа вершин графа Γ, которые достаточно исследовать для ответа на вопрос об $\mathcal{F}$-эквивалентности программ π_1, π_2. Эффект подавления $æ_s$, индуцированный состоянием данных s, — это функция преобразования состояний данных следующего вида: $æ_s(s_1) = s_2$, если $s_1 \circ s = s_2 \circ s$ и при этом состояние s_2 выбрано так, что наименьшая длина принадлежащих этому состоянию цепочек операторов имеет максимально возможное значение. В разделе формулируются и обосновываются свойства эффектов подавлений, включая конечность множества всевозможных эффектов подавления и его частичную упорядоченность относительно преобразования состояний данных операторами. Кроме того, в разделе доказано существование и приводится явное описание автомата, распознающего эффекты подавления: на вход автомату подаются цепочки операторов; эффекты подавления $æ_{[\![h]\!]}$, $æ_{[\![g]\!]}$ совпадают тогда и только тогда, когда совпадают результаты работы автомата на цепочках h, g; если $[\![h]\!] = [\![g]\!]$, то цепочки h, g приводят автомат в одно состояние данных. Описание автомата, распознающего эффекты подавления, основано на описании автомата, распознающего подавление операторов.

Также в этом разделе вводятся и исследуются два обобщения понятия эффекта подавления: развитие и обобщённое развитие эффекта подавления. Содержательно, эти развития позволяют отслеживать изменение эффектов подавления при построении вычислений программ. С использованием конечности и частичной упорядоченности эффектов подавления доказывается, что существует лишь конечное число способов изменения эффекта подавления при построении вычислений программы, то есть лишь конечное число развитий и обобщённых развитий. Развития и обобщённые развития вместе с автоматом,

распознающим эффекты подавления, позволяют разбить вершины графа Γ на полиномиальное относительно размеров программ число групп так, что по накоплении полиномиального числа вершин в группе при обходе графа Γ можно либо констатировать существование в этом графе опровергающего маршрута (и следовательно, неэквивалентность программ), либо игнорировать остальные вершины группы в дальнейшем обходе. Такое разбиение обеспечивается леммами, сформулированными в *разделе 5.6*. Алгоритм проверки $\mathcal{F}$-эквивалентности программ описан в *разделе 5.7* и состоит в обходе графа совместных вычислений: обход начинается в корне графа; если в одной из групп исследовано достаточно много вершин, то в зависимости от свойств этой группы либо констатируется неэквивалентность программ, либо остальные вершины группы игнорируются в дальнейшем обходе; если граф совместных вычислений полностью построен, то эквивалентность программ определяется отсутствием в нём опровергающих маршрутов. Корректность и полиномиальность алгоритма строго обосновываются, что позволяет считать доказанной основную теорему главы 5, которой завершаются раздел и глава.

Теорема 5.7.1. *Пусть $\mathcal{F}$ — упорядоченная шкала с коммутативностью и подавлением. Тогда проблема $\mathcal{F}$-эквивалентности пропозициональных последовательных программ разрешима за время, полиномиальное относительно размеров программ.*

В шестой главе исследуется проблема эквивалентности линейных унарных рекурсивных программ на упорядоченных полугрупповых шкалах. Результаты исследования следующие. Устанавливается набор требований к упорядоченной полугрупповой шкале $\mathcal{F}$, достаточных для описания графа совместных вычислений $\Gamma = \Gamma(\pi_1, \pi_2, \mathcal{F})$ линейных унарных рекурсивных программ π_1, π_2 и полиномиального алгоритма проверки $\mathcal{F}$-эквивалентности программ π_1, π_2, основанного на обходе этого графа. В предположении о том, что шкала $\mathcal{F}$ удовлетворяет установленным требованиям, обосновываются корректность и сложность алгоритма проверки эквивалентности.

Одно из требований, предъявляемых к шкале $\mathcal{F}$, приведено в *разделе 6.1* и состоит в наличии критериальной системы для этой шкалы. На основе критериальной системы вводится удобное для исследования различие пар линейных термов. Критериальная система может быть определена так. Рассмотрим линейные термы t_1, t_2. Опустим символы $t_1.\phi$, tt_2 — они в явном виде сохраняются в различии термов. Оставшиеся цепочки операторов $t_i.l$, $t_i.r$ заменим состояниями данных $s_i = [\![t_i.l]\!]$, $s'_i = [\![t_i.r]\!]$. Выберем моноид U, элементы моноида объявим различиями состояний данных. Отобразим состояния s_1, s_2 в моноид U гомоморфизмом $\varphi : \mathcal{F}.S \times \mathcal{F}.S \to U$. То же сделаем для элементов s'_1, s'_2. Расширим моноид U до моноида W (далее он называется критериальным), добавив элементы, отмечающие начало и конец совместной работы программ: w^+ и w^* соответственно. Пусть $\circ$ — операция моноида W. Помимо символов $t_1.\phi$, tt_2 в различии термов сохраняются элементы $w^+ \circ \varphi(s_1, s_2)$ и $\varphi(s'_1, s'_2) \circ w^*$. Моноид W с выделенными элементами w^+, w^*, подмоноидом U и гомоморфизмом φ образуют критериальную систему.

Символом e обозначается нейтральный элемент критериального моноида. На критериальную систему для шкалы $\mathcal{F}$ налагаются следующие требования:

1. равенство $w^+ \circ \varphi(s_1, s_2) \circ w^* = e$ верно тогда и только тогда, когда $s_1 = s_2$;
2. любой элемент класса смежности $w^+ \circ U$ имеет не более одного правого обратного элемента в классе смежности $U \circ w^*$;
3. любой элемент класса смежности $U \circ w^*$ имеет не более одного левого обратного элемента в классе смежности $w^+ \circ U$.

Первое требование к критериальной системе позволяет определить по различию термов, которыми оканчиваются вычисления программ, совпадают ли результаты этих вычислений. Остальные требования позволяют ограничить число вершин, которое достаточно исследовать в графе совместных вычислений

$\Gamma = \Gamma(\pi_1, \pi_2, \mathcal{F})$ для определения $\mathcal{F}$-эквивалентности программ π_1, π_2. Описание графа совместных вычислений приводится в *разделе 6.2* и основывается на приведённом выше понятии различия линейных термов и на следующей стратегии совместного выполнения. Пусть при совместном выполнении программами π_1, π_2 получены соответственно термы t_1, t_2. Если $[\![t_i.l]\!] \prec [\![t_{3-i}.l]\!]$, то активна только программа π_i. Иначе активны обе программы. При этом если активная программа завершила построение вычисления, то она объявляется неактивной.

Опровергающие маршруты графа совместных вычислений могут быть описаны в терминах совместного выполнения следующим образом. Случай 1: в результате совместного выполнения программами построены конечные вычисления с различными результатами. Случай 2: совместное выполнение бесконечно, программа π_i почти всегда неактивна, при этом символы $t.\phi$ всех получаемых ей термов t завершаемы ($i \in \{1, 2\}$). В разделе 6.2 формулируется и обосновывается утверждение, позволяющее свести проверку $\mathcal{F}$-эквивалентности нормализованных линейных унарных рекурсивных программ π_1, π_2 к проверке отсутствия опровергающих маршрутов в графе Γ. В формулировке утверждения предполагается наличие критериальной системы для шкалы $\mathcal{F}$, то есть возможность описания графа Γ.

Теорема 6.2.1. *Программы $\mathcal{F}$-эквивалентны тогда и только тогда, когда в графе Γ отсутствуют опровергающие маршруты.*

В *разделе 6.3* приводятся утверждения, позволяющие ограничить число вершин графа совместных вычислений, которые достаточно исследовать для определения $\mathcal{F}$-эквивалентности программ. На основании этого факта описывается алгоритм, решающий проблему $\mathcal{F}$-эквивалентности программ. Алгоритм состоит в нормализации программ и обходе графа совместных вычислений получившихся нормализованных программ: обход начинается в корне графа; в процессе обхода сохраняется список всех посещённых вершин; если накоплено достаточно много вершин, то программы признаются неэквивалентными; если граф совместных вычислений полностью исследован, то эквивалентность про-

грамм определяется отсутствием в нём опровергающих маршрутов. В своём описании алгоритм использует два дополнительных предположения. Предположение 1: предоставлен алгоритм, принимающий на вход цепочки операторов h_1, h_2 и проверяющий соотношение $[\![h_1]\!] \prec [\![h_2]\!]$. Предположение 2: предоставлен алгоритм, проверяющий равенство элементов критериального моноида в классах смежности $w^+ \circ U$, $U \circ w^*$ и $w^+ \circ U \circ w^*$; на вход алгоритму подаются цепочки операторов h_1, h_2, и ими описывается элемент $\varphi([\![h_1]\!], [\![h_2]\!])$ моноида U. Задачи, сформулированные в последних двух предположениях, далее называются соответственно задачей достижимости и задачей равенства. Основной результат раздела 6.3 может быть сформулирован следующим образом.

Теорема 6.3.1. *Если существует критериальная система для упорядоченной полугрупповой шкалы $\mathcal{F}$ и задачи равенства и достижимости разрешимы, то проблема $\mathcal{F}$-эквивалентности линейных унарных рекурсивных программ разрешима.*

Из утверждений раздела 6.3 следует в худшем случае экспоненциальная относительно размеров программ оценка числа исследуемых вершин графа совместных вычислений. Эта оценка снижается до полиномиальной утверждениями, сформулированными и доказанными в *разделе 6.4* и использующими дополнительное предположение о том, что критериальный моноид является группой. В разделе 6.4 предложен полиномиальный алгоритм проверки $\mathcal{F}$-эквивалентности линейных унарных рекурсивных программ. Этот алгоритм может быть получен из алгоритма в разделе 6.3 следующими изменениями. В задачу равенства добавляется следующая подзадача: проверить равенство элементов $w_1^{-1} \circ w_2^{-1} = w_3^{-1} \circ w_4^{-1}$, где w_1 и w_3 — элементы класса $w^+ \circ U$, а w_2, w_4 — элементы класса $U \circ w^*$; способ задания элементов классов смежности остаётся тем же, что и ранее в задаче равенства. Кроме того, предполагается полиномиальная разрешимость задач достижимости и равенства. Корректность и полиномиальность алгоритма строго обосновываются, что позволяет считать доказанной основную теорему главы 6, которой завершаются раздел и глава.

Теорема 6.4.1. *Пусть: $\mathcal{F}$ — упорядоченная полугрупповая шкала; для этой шкалы существует критериальная система; критериальный моноид этой системы является группой; задачи достижимости и равенства полиномиально разрешимы. Тогда проблема $\mathcal{F}$-эквивалентности линейных унарных рекурсивных программ разрешима за время, полиномиальное относительно размеров программ.*

В седьмой главе доказывается полиномиальная разрешимость проблемы эквивалентности металинейных унарных рекурсивных программ на свободной шкале. Далее рассматриваются свободная шкала $\mathcal{F}$ и металинейные унарные рекурсивные программы π_1, π_2. Без ограничения общности эти программы считаются нормализованными. Конечное вычисление нормализованной программы обязательно оканчивается цепочкой операторов.

В *разделе 7.1* вводится понятие веса $\|\pi\|_t$ терма t в программе π: рассматриваются все вычисления программы π', получаемой из π заменой запроса на терм t; если все такие вычисления бесконечны, то вес бесконечен; иначе выбирается вычисление, оканчивающееся наименьшей по длине цепочкой операторов, и весом объявляется длина этой цепочки. Также в разделе приводятся и обосновываются утверждения, позволяющие считать, что запросы исследуемых программ не содержат тупикового символа и имеют одинаковый вес (в своих программах).

В *разделе 7.2* описывается граф совместных вычислений $\Gamma = \Gamma(\pi_1, \pi_2, \mathcal{F})$ нормализованных металинейных унарных рекурсивных программ π_1, π_2. Различия пар термов, а также стратегия совместного выполнения описываются следующим образом. Пусть в процессе совместного выполнения программами получены термы t_1, t_2.

Случай 1: $t_1.l = t_2.l$, и ни один из термов t_1, t_2 не содержит вхождений тупикового символа. Различие термов t_1, t_2 схематично изображено на рисунке 3. Левым прямоугольником изображается символ $t_i.\phi$, овалом — цепочка операторов $t_i.r.l$, правым прямоугольником — последовательность заголовков $t_i.r.r$.

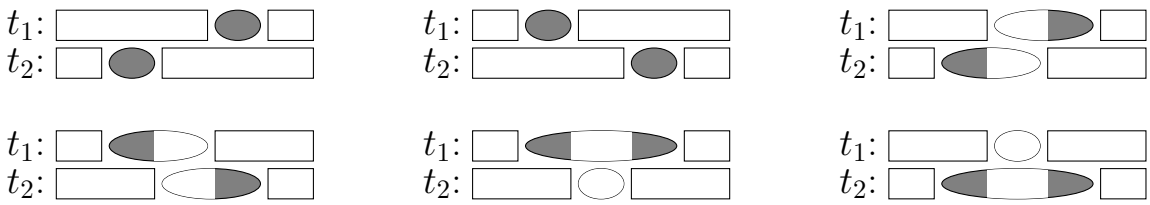


Рис. 3. Различие термов. Случай 1

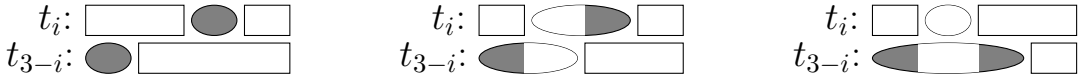


Рис. 4. Различие термов. Случай 2

Ширина фигур пропорциональна весу изображаемых ими термов. В различии сохраняются термы, изображаемые прямугольниками, и цепочки операторов, выделенные серым цветом. В условие данного случая включается требование: цепочки операторов, отвечающие белым частям овалов, равны. Обе программы в этом случае активны.

Случай 2: $t_i.l \prec t_{3-i}.l$, и ни один из термов t_1, t_2 не содержит вхождений тупикового символа. Различие термов t_1, t_2 схематично изображено на рисунке 4. Овалом для t_{3-i} изображается цепочка операторов $t_{3-i}.l$, из которой вычеркнут префикс длины $|t_i.l|$, а прямоугольником — терм $t_{3-i}.\phi\ t_{3-i}.r$. В остальном смысл изображений и требование равенства белых частей овалов те же, что и в случае 1. В этом случае активна только программа π_i.

Случай 3: $t_1 = t_2 \in \mathfrak{O}^*$. Таким термам сопоставляется особое различие, означающее, что программами построены конечные вычисления с равными результатами. Обе программы неактивны.

Случай 4: в каждый из термов t_1, t_2 входит тупиковый символ. Таким термам сопоставляется различие, означающее невозможность обеих программ завершить свою работу. Обе программы неактивны.

Случай 5: ни одно из условий предыдущих случаев не выполнено. Таким термам сопоставляется различие, выраженное особой вершиной графа Г, на-

зываемой далее опровергающей. Смысл этой вершины пояснён далее. Обе программы неактивны.

Опровергающими объявляются маршруты графа Г, начинающиеся в корне и оканчивающиеся в опровергающей вершине. В разделе 7.2 формулируется и обосновывается утверждение, позволяющее свести проверку $\mathcal{F}$-эквивалентности нормализованных металинейных унарных рекурсивных программ к проверке наличия опровергающих маршрутов в графе Г.

Теорема 7.2.1. *Нормализованные металинейные унарные рекурсивные программы π_1, π_2 эквивалентны на свободной шкале $\mathcal{F}$ тогда и только тогда, когда в графе $\Gamma(\pi_1, \pi_2, \mathcal{F})$ из корня не достижима опровергающая вершина.*

В *разделе 7.3* формулируются и обосновываются утверждения, из которых следует, что число вершин графа совместных вычислений эквивалентных программ полиномиально относительно размеров программ. Алгоритм проверки эквивалентности приведён в *разделе 7.4* и состоит в нормализации программ и обходе графа совместных вычислений получившихся нормализованных программ: обход начинается в корне; если обходом исследовано достаточно много вершин, то программы признаются неэквивалентными; если граф совместных вычислений полностью построен, то эквивалентность программ определяется отсутствием в нём опровергающих маршрутов. Корректность и полиномиальность алгоритма строго обосновываются, что позволяет считать доказанной основную теорему главы 7, которой завершаются раздел и последняя глава основной части диссертации.

Теорема 7.4.1. *Проблема эквивалентности металинейных унарных рекурсивных программ на свободной шкале разрешима за время, полиномиальное относительно размеров программ.*

В восьмой главе — заключении — перечисляются основные результаты работы и обсуждается значимость этих результатов.

Список публикаций

1. Захаров В. А., Подымов В. В. Об одной полугрупповой модели программ, определяемой при помощи двухленточных автоматов // Научные ведомости Белгородского государственного университета. Серия История, экономика, политология, информатика. — 2010. — Т. 14, № 7. — С. 94–101.
2. Подымов В. В. Алгоритм проверки эквивалентности линейных унарных рекурсивных программ на упорядоченных полугрупповых шкалах // Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. — 2012. — № 4. — С. 37–43.
3. Подымов В. В. О проверке сильной эквивалентности металинейных унарных рекурсивных программ // Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. — 2013. — № 1. — С. 21–27.
4. Подымов В. В., Захаров В. А. Полиномиальный алгоритм проверки эквивалентности в модели программ с перестановочными и подавляемыми операторами // Труды Института системного программирования РАН. — 2014. — Т. 26, вып. 3. — С. 145–166.
5. Захаров В. А., Подымов В. В. Об эквивалентности металинейных унарных рекурсивных программ // Материалы XI Международного семинара “Дискретная математика и ее приложения”, посвященного 80-летию со дня рождения академика О. Б. Лупанова. — 2012. — С. 157–159.
6. Подымов В. В. О проверке эквивалентности последовательных и рекурсивных программ на упорядоченных полугрупповых шкалах // Материалы X Международной конференции “Интеллектуальные системы и компьютерные науки”. — 2011. — С. 295–298.
7. Подымов В. В. Быстрый алгоритм проверки эквивалентности программ с коммутативными и подавляемыми операторами // Материалы XVII международной конференции “Проблемы теоретической кибернетики” — 2014. — С. 234–237.