На правах рукописи

Нечесов Андрей Витальевич

Полиномиальная вычислимость в семантическом программировании

Специальность 1.1.5 —
«Математическая логика, алгебра, теория чисел
и дискретная математика.»

Автореферат
диссертации на соискание учёной степени
кандидата физико-математических наук

Новосибирск — 2022

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институт математики им. С. Л. Соболева Сибирского отделения Российской академии наук.

Научный руководитель: академик РАН, д.ф.-м.н., профессор
Гончаров Сергей Савостьянович

Официальные оппоненты: **Свириденко Дмитрий Иванович**,
д.ф.-м.н., доцент,
Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики», Новосибирск,
профессор кафедры САПР

Манцивода Андрей Валерьевич,
д.ф.-м.н., профессор,
Федеральное государственное бюджетное образовательное учреждение высшего образования «Иркутский государственный университет», Иркутск,
руководитель исследовательского центра интеллектуального управления контентом ИГУ

Ведущая организация: Федеральное государственное автономное образовательное учреждение высшего образования "Казанский (Приволжский) федеральный университет".

Защита состоится 31 марта 2023 г. в 15 часов 30 минут

на заседании диссертационного совета 24.1.074.02 при Федеральном государственном бюджетном учреждении науки Институт математики им. С. Л. Соболева Сибирского отделения Российской академии наук по адресу: 630090, Новосибирск, проспект академика Коптюга, д. 4.
С диссертацией можно ознакомиться в библиотеке Федерального государственного бюджетного учреждения науки Институт математики им. С. Л. Соболева Сибирского отделения Российской академии наук и на сайте http://math.nsc.ru.

Автореферат разослан __ __ ____ года.

Ученый секретарь
диссертационного совета
24.1.074.02,
д.ф.-м.н. Дудкин Федор Анатольевич

Постановка задачи и актуальность темы диссертации.

В 21 веке информационные технологии сделали серьезный скачок в своем развитии. Бурными темпами начали развиваться такие направления как криптовалюты, блокчейны, умные контракты, децентрализованные системы, нейронные сети, искусственный интеллект, робототехника и т.д. Во всех этих направлениях исследований требуется, чтобы все процессы происходили быстро и четко, без задержек и зацикливаний. Одной из важнейших характеристик для этого является понятие полиномиальной временной сложности. Т.е. процесс вычисления и выдача результата должны быть осуществлены не позднее чем через время равное некоторому полиному от длины входных данных. Далеко не все алгоритмы на сегодняшний день удовлетворяют этой характеристике.

Проблема 1. *Проблема быстродействия, целостности, адаптивности и надежности интеллектуальных систем. [5]*

Но порой одной полиномиальной временной сложности недостаточно, особенно, если это касается направления искусственного интеллекта [33]. Где нужно не только быстро выдать конечный результат, но и объяснить человеку на понятном ему языке, почему был выдан этот результат. К сожалению, большинство реализованных на сегодняшний день алгоритмов не могут этого сделать или делают это крайне плохо. Почти все они реализованы с помощью нейронных сетей [28] или других алгоритмов машинного обучения [21], которые натренированы на определенных данных и могут выдавать только конечный результат и то с достаточно большой долей погрешности. Эти сети не могут логически объяснить человеку, почему этот результат был выбран. Поэтому такого типа интеллектуальные системы чаще всего не могут быть использованы на передовых и ответственных направлениях таких как: проведение серьезных хирургических операций, управление роботами, управление автомобилями и т.д.

Возникает так называемая проблема черного ящика в искусственном интеллекте. Необходимо не просто выдать результат быстро и правильно, но еще логически обосновать его. Поэтому второй характеристикой для всех этих процессов служит объяснительная составляющая.

Проблема 2. *Проблема черного ящика в ИИ. [18]*

Для решения этой проблемы в ИИ было представлено направление объяснительного искусственного интеллекта (далее XAI). Интеллектуальные системы на основе XAI не просто выдают результат, но еще и объясняют человекопонятным языком как этот результат получен. Для этих целей как нельзя лучше подходят высокоуровневые языки программирования в которых видна логика исполнения программы. Особое место среди

этих языков занимают логические языки программирования базирующиеся на основных конструкциях математической логики, таких как формулы и термы.

Проблема 3. *Проблема p-полноты для языков программирования. [36]*

Большинство высокоуровневых языков программирования (Паскаль, Си++) [15], в том числе и логических языков программирования (PROLOG [34], Libretto [13] и т.д.) являются Тьюринг полными [30]. Другая часть языков, наоборот, являются очень сильно урезанными и программы на них выполняются либо за линейное время, либо за полиномиальное время, где степень полинома крайне низкая. Основная проблема p-полноты - это создать язык программирования такой, что любая программа, реализованная в этом языке, будет исполняться за полиномиальное время от длины входных данных. И в то же время любой полиномиальный алгоритм может быть реализован при помощи некоторой программы этого языка.

Концепция семантического программирования

В 70-х - 80-х годах прошлого столетия академиками Ю.Л.Ершовым и С.С.Гончаровым, а также профессором Д.И.Свириденко была разработана концепция семантического программирования. Основная идея заключалась в разработке логического языка программирования с использованием основных конструкций математической логики, таких как формулы и термы. В качестве исполняющего устройства выступала наследственно-конечная списочная надстройка $HW(\mathfrak{M})$ сигнатуры σ [22;23;25], в которой все функции и предикаты вычислимы. А в качестве Σ-программ выступали синтаксические конструкции задаваемые с помощью Σ-определений вида [8]:

$$\underline{def}\ Q(\overline{x}) :\ \Phi(\overline{x},Q,F)$$

а также вида

$$\underline{def}\ F(x_1,\ldots,x_{n-1}) = x_n :\ \Phi(\overline{x},Q,F)$$

где Q и F - наборы позитивно входящих в Φ предикатных и функциональных переменных. При этом в последнем случае формула Φ определяет функциональное равенство. Но построенный логический язык программирования в рамках концепции семантического программирования являлся Тьюринг полным и не решал проблему p-полноты.

В рамках семантического программирования уже реализованы несколько языков программирования таких как: d0sl [20], sDSL [27], Libretto [13] и язык документных моделей [12;14]. Но основная проблема, что ни для одного из них не доказана p-полнота. Более того, Libretto

вообще является Тьюринг полным языком. Поэтому стояла задача, как в рамках концепции семантического программирования найти такие ограничения, чтобы получить язык со свойством р-полноты.

Проблема 4. *Как в рамках семантического программирования увеличить выразительную силу языка и найти подходящие синтаксические конструкции, при которых вычислительная сложность логических программ не выходила бы за рамки полиномиальной и при этом любой алгоритм полиномиальной сложности мог бы быть реализован подходящей программой в этом языке?*

Для ответа на первую часть вопроса такие синтаксические конструкции были найдены Гончаровым. Он предложил рассматривать только формулы с ограниченными кванторами определенного вида (Δ_0-формулы), а также ввел понятие условного терма [6].

Далее последовал ряд работ в которых были введены другие типы термальных расширений: рекурсивные термы, р-рекурсивные термы, ограниченные Δ_0-термы и т.д. [7; 24; 26]. Кроме рекурсивных термов, термальные расширения с помощью остальных перечисленных термов не выводят за рамки полиномиальности [7]. Более того, понятие логической программы также изменилось, если раньше ключевым элементом логической программы была Σ-формула (Δ_0-формула), то теперь под программой стал пониматься некоторый специальный терм. Понятие специального терма индуктивно задается с помощью термального расширения стандартных термов условными термами, р-рекурсивными термами и другими видами термов.

В работе [31] было показано, что сложность проверки истинности любой Δ_0-формулы и сложность вычисления любого терма является полиномиальной в р-вычислимой наследственно-конечной списочной надстройке $HW(\mathfrak{M})$ сигнатуры σ. Так как понятие терма пришлось индуктивно расширить, то автоматически расширилось и понятие Δ_0-формулы. Уже при этих термальных расширениях возник вопрос, а все ли полиномиальные алгоритмы можно реализовать с помощью этих программ? Этот вопрос оставался открытым несколько лет.

Только в 2021 году в нашей совместной работе [36] с Гончаровым удалось построить необходимую синтаксическую конструкцию р-итерационного терма и термально расширить существующий язык L_1. В полученном языке L вычислительная сложность [29; 32] каждой логической программы является полиномиальной и, более того, для любого полиномиального алгоритма существует подходящая L-программа реализующая его. Это удалось достичь, благодаря моделированию работы машины Тьюринга на первых $h(|x|)$ тактах с помощью р-итерационного терма, где $h(|x|)$ – некоторый многочлен от длины входных данных.

Еще одним важным шагом было понимание того, что с помощью списочных элементов $HW(M)$ можно кодировать индуктивно задаваемые объекты любой природы. Выделять же эти элементы будут некоторые новые одноместные предикаты. При этом новые элементы могут индуктивно формульно определяться через базовые элементы основного множества первоначальной модели. Если перенестись в плоскость высокоуровневых языков программирования, то индуктивно определяемыми элементами через базовые элементы 0 и 1, могут выступать натуральные числа, наследственно-конечные списки, массивы, матрицы, слова, предложения и т.д. И для того чтобы правильно определить такие конструкции и не выйти за рамки полиномиальной сложности нам нужны обоснованные математические оценки.

Одним из важнейших результатов в теории Σ-определимости [10] является классическая теорема Ганди [10;17], в которой по любой Σ-формуле, индуктивно расширяющей предикат, строится специальный оператор, наименьшая неподвижная точка которого Σ-определима. Если мы хотим применить этот результат в семантическом программировании, то здесь возникает вопрос, как для новых типов элементов, индуктивно задаваемых через базовые элементы основного множества модели, применить классическую теорему Ганди?

На все вышепоставленные вопросы в этой работе даются ответы. Более того, были исследованы вопросы связанные с существованием полиномиально вычислимых представлений для множества выводов в порождающих грамматиках [16; 39], которые играют ключевую роль в построении синтаксиса различных лингвистических языков, а также для построения языков программирования таких как Pascal, C++, PHP, Solidity, Java и т.д. Очень важно, чтобы существовали быстрые алгоритмы проверки, является ли некоторый программный код программой в соответствующем языке и какова вычислительная сложность этой проверки.

Далее, встал вопрос о сложности распознавания некоторых синтаксических объектов из логики предикатов первого порядка. К этим объектам относятся термы, формулы, линейные доказательства и доказательства в виде дерева из логики предикатов первого порядка. Более того, встал вопрос о существовании полиномиально вычислимых представлений для L-программ и L-формул построенного нами языка L.

В диссертации получены результаты, которые дают ответы на вышепоставленные вопросы. Более того, исследован достаточно большой класс множеств математических объектов, для которых существуют полиномиально вычислимые списочные представления. Это позволяет использовать данные синтаксические конструкции в p-полных языках программирования без увеличения вычислительной сложности программ.

Особенно хотелось бы отметить результат о существовании полиномиально вычислимого списочного представления для множества доказательств. Данный результат позволяет внедрять проверку действий того

или иного интеллектуального устройства в рамках заданных начальных понятий и логических правил. Корректность и человекопонятность рассуждений является важной составляющей для объяснительного искусственного интеллекта, а также для робототехники.

Основные результаты диссертации.

1. Решена проблема равенства классов P и L.
2. Построен полиномиальный аналог классической теоремы Ганди о наименьшей неподвижной точке с начальными p-вычислимыми условиями.
3. Получена серия результатов о существовании полиномиально вычислимых представлений:
   - для множества термов и множества формул ИП.
   - для множества L-программ и множества L-формул.
   - для множества доказательств ИП в виде дерева.
   - для множества линейных доказательств ИП.
   - для множества выводов в порождающих грамматиках.

Результаты пунктов $1-2$ доказаны в соавторстве со своим научным руководителем С.С. Гончаровым [35; 36]. Результаты пункта 3 получены автором диссертации лично [37;38].

Вклад соавтора в основной результат 1: в работе [24] Гончаровым и Свириденко были введены p-рекурсивные термы, а также, немного ранее, Гончаровым было введено понятие условного терма в [6]. Термальные расширения первоначального логического языка этими конструкциями не выводили за рамки полиномиальности. Далее, Гончаровым была поставлена проблема p-полноты полученного языка. Это решение было найдено Гончаровым и Нечесовым (автором диссертации) в работе [36]. Гончаровым было предложено построить терм прямо по машине Тьюринга, вычисляющей некоторую полиномиально вычислимую функцию. Нечесовым была найдена конструкция p-итерационного терма, являющаяся специальным случаем ограниченной рекурсии, термальное расширение с помощью которой оказалось достаточным для реализации всех полиномиально вычислимых функций. Нечесовым также было доказано, что данное термальное расширение L не выводит за рамки полиномиальности. А в силу того, что любая полиномиально вычислимая функция реализуется с помощью подходящей L-программы, то отсюда автоматически вытекает равенство классов P и L.

Вклад соавтора в основной результат 2: Нечесовым было предложено построить полиномиальный аналог классической теоремы Ганди, который бы мог единообразно применяться к индуктивно задаваемым конструкциям. Данные синтаксические конструкции появлялись в работах Гончарова и его соавторов [6; 7; 24; 26]. Нечесовым были построены полиномиально вычислимые порождающие семейства формул, но доказательство в общем

случае не проходило. В результате совместных обсуждений было выработано более точное построение локального шага оператора замыкания, которое и позволило Нечесову получить окончательное доказательство полиномиально аналога теоремы Ганди о наименьшей неподвижной точке.

Новизна и научная значимость. В работе все полученные результаты новые. Работа носит теоретико-практический характер. Результаты работы могут быть использованы как для дальнейших исследований, так и для реализации методологии построения программ в высокоуровневых языках, базирующейся на теории семантического программирования.

Построенный в работе логический язык программирования L является первым p-полным высокоуровневым языком. До этого момента либо высокоуровневые языки программирования были Тьюринг полными, либо имели сильные ограничения, что не позволяло реализовать ряд алгоритмов полиномиальной вычислительной сложности. Новый же язык логического программирования L помимо того, что является p-полным, также имеет сильные выразительные свойства, что позволяет использовать его в алгоритмах объяснительного искусственного интеллекта. Более того, в качестве базовых библиотек могут быть использованы модули полиномиальной вычислительной сложности реализующие работу полносвязных нейронных сетей прямого распространения с полиномиально вычислимыми функциями активации, модули реализующие построение элементов с помощью порождающих грамматик и их производных.

Доказанный в работе полиномиальный аналог теоремы Ганди дал новый способ доказательства полиномиальной вычислимости для индуктивно задаваемых синтаксических конструкций с помощью построения подходящих p-вычислимых GNF-систем. Как оказалось, в процессе изучения этого направления, свойство индуктивного задания через вновь определяемые элементы присуще очень многим множествам объектов. Наиболее важными из них являются списки, массивы, компьютерные программы, логические формулы и термы, доказательства в исчислении предикатов первого порядка и т.д. Поэтому применение данной теоремы может сильно облегчить понимание того какие индуктивные конструкции следует использовать, а какие нет.

Еще одним важным результатом является разработка нового метода, основанного на использовании PAG-теоремы, p-итерационных термов и свойств L-программ и L-формул. Данный метод позволяет показывать полиномиальную вычислительную сложность множеств и алгебраических структур не прибегая к построению машин Тьюринга и подсчету числа тактов работы. Для этого достаточно построить подходящую L-программу или подходящую L-формулу.

Методы исследования. В работе используются классические методы теории моделей, теории алгоритмов, базовые методы семантического

программирования, методы формульной определимости и теории сложности.

Апробация работы. Результаты диссертации были представлены на конференциях:

- международная конференция «Мальцевские чтения» Новосибирск, 2019, 2022
- международная конференция: «IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)», 2022
- «Global Summit on Robotics and Artificial Intelligence», Prague, 2022
- международная школа-семинар «Синтаксис и семантика логических систем». Владивосток, 2022
- международная конференция «Актуальные задачи математики, механики и информатики», Караганда, 2022
- международная конференция «Current State and Development Perspectives of Digital Technologies and Artificial Intelligence», Самарканд, 2022
- международная конференция «Mathematical Logic and Computer Science», Астана, 2022

Также результаты работы докладывались на семинаре «Теория вычислимости» в 2021 году в Институте математики им. С. Л. Соболева СО РАН и в 2022 году на математическом семинаре в Иркутском Государственном Университете.

Публикации. Основные результаты автора по теме диссертации опубликованы в 4-х работах [35–38] из журналов списка ВАК. Из них 2 работы [35;36] опубликованы в соавторстве со своим научным руководителем С.С. Гончаровым и 2 работы [37;38] опубликованы автором лично.

Работа автора [39] из журнала списка ВАК является англоязычной версией работы автора [37].

Более того, по тематике диссертации у автора есть 2 работы [40;41] в сборниках конференций не из списка ВАК.

Объем и структура работы. Диссертация состоит из введения, 4 глав и заключения. Полный объём диссертации составляет 80 страниц. Список литературы содержит 41 наименование.

Общее содержание диссертации

Общая структура диссертации.

Диссертация разбита на главы, которые в свою очередь разбиваются на вступительную часть и параграфы. В каждой главе для лучшего ознакомления даются вспомогательные сведения и определения. Теоремы, Утверждения и Леммы имеют двойную нумерацию, где первое число – это номер главы, а второе – это номер уже самого утверждения (леммы, теоремы и т.д.). Замечания же имеют тройную нумерацию, где помимо главы указан и номер параграфа.

Во **введении** содержится постановка задачи, обосновывается актуальность темы исследования, освещается степень ее разработки; изложены цели, задачи, методы исследования и основные результаты диссертации; отражены новизна и научная значимость работы и данные об апробации. Также приведены сведения о публикации результатов диссертации.

Первая глава посвящена решению проблемы построения такого логического языка программирования L, который бы совпадал с классом P всех алгоритмов полиномиальной вычислительной сложности. Под равенством классов P и L подразумевается, что для каждого алгоритма из класса P существует программа из класса L, реализующая его, а также то, что вычислительная сложность любой программы из класса L является полиномиальной. Для доказательства этого утверждения в рамках семантического программирования были введены ряд вспомогательных синтаксических термальных конструкций, а также был изменен взгляд на то, что считать программой. Если раньше в основе понятия логическая программа лежало понятие Δ_0-формулы, то теперь в этой роли стал выступать терм специального вида. Но сами Δ_0-формулы играют непосредственную роль в построении этих термов. Например, в определении условного терма [6], а также p-итерационного терма [36] Δ_0-формулы участвуют в построении напрямую.

Первый параграф посвящен базовым определениям, в том числе дается определение полиномиально вычислимой функции, полиномиально вычислимого множества и других конструкций таких как: полиномиально вычислимые неподвижные точки, модели и т.д [1–4]. Для краткости такие конструкции будем называть p-вычислимыми [19; 35]. Также дается понятие полиномиально вычислимого представления (p-вычислимого представления) для произвольного множества объектов. Одним из эффективных представлений таких множеств является списочное представление. Как нельзя лучше для этих целей подошла p-вычислимая наследственно-конечная списочная надстройка $HW(\mathfrak{M})$ сигнатуры σ [31]. В основном множестве этой модели с помощью списочных конструкций достаточно легко кодируются p-вычислимые множества объектов различного вида. Более того, $HW(\mathfrak{M})$ сигнатуры σ выступает в качестве исполняющего устройства

в теории семантического программирования. Все результаты диссертации получены в рамках этой теории. Чтобы успешно доказывать р-вычислимость различных списочных представлений нам понадобится обогащение модели различными функциями, предикатами и константами. Для этих целей определяются сигнатуры: σ_N – для работы с натуральными числами, σ_W – для работы со списками, σ_S – вспомогательные операции и константные символы, σ_{ext} – множество константных символов для внешней сигнатуры σ^{ext}, σ_l – множество константных символов для логических связок, кванторов, символа d для выделения переменных в формулах сигнатуры σ_{ext}, σ_L – для кодирования L-формул и L-программ сигнатуры σ. Без ограничения общности считаем, что первые три набора сигнатурных символов включены в σ по умолчанию. Остальные добавляются в случае, если мы работаем с синтаксическими объектами некоторой фиксированной внешней сигнатуры σ^{ext}. При этом в силу полиномиальной вычислимости всех этих предикатных и функциональных конструкций р-вычислимость модели $HW(\mathfrak{M})$ расширенной сигнатуры σ сохраняется.

Во *втором параграфе* дается определение условного терма. В 2017 году, Гончаров в своей работе [6] представил первую логическую программу, которая основывалась на индуктивном понятии условного терма. Расширение L_1 является термальным расширением языка L_0 с помощью условных термов. Если в L_0 термами были стандартные термы логики предикатов первого порядка, а формулами – стандартные Δ_0-формулы с ограниченными кванторами по отношениям $\in$, $\subseteq$, $\leq$ сигнатуры σ, то в L_1 в построении термов и формул могут участвовать условные термы. Конструкция условного терма является естественным аналогом оператора «if then else» из высокоуровневых языков программирования. Оспичевым и Пономаревым в их работе [31] было показано, что в р-вычислимой модели $HW(\mathfrak{M})$ конечной сигнатуры σ вычислительная сложность любой L_1-программы (терма, в построении которого могут участвовать условные термы) и сложность проверки истинности любой L_1-формулы (формулы, в построении которой могут участвовать условные термы) являются полиномиальными.

Сама же синтаксическая структура условного терма имеет вид:

$$t(\overline{v}) = \begin{cases} t_0(\overline{v}), \text{ если } HW(\mathfrak{M}) \models \varphi_0(\overline{v}) \\ t_1(\overline{v}), \text{ если } HW(\mathfrak{M}) \models \varphi_1(\overline{v}) \& \neg\varphi_0(\overline{v}) \\ \dots \\ t_n(\overline{v}), \text{ если } HW(\mathfrak{M}) \models \varphi_n(\overline{v}) \& \neg\varphi_0(\overline{v}) \& \dots \& \neg\varphi_{n-1}(\overline{v}) \\ t_{n+1}(\overline{v}), \text{ иначе} \end{cases} \tag{1.1}$$

где t_i – L_1-программы, φ_j – L_1-формулы.

С точностью до переобозначений, введенных в данной работе, можно сформулировать следующее утверждение.

Утверждение 1.1. *(Оспичев, Пономарев) [31] Вычислительная сложность любой L_1 - программы и сложность проверки истинности любой L_1-формулы в р-вычислимой наследственно-конечной списочной надстройке $HW(\mathfrak{M})$ конечной сигнатуры σ является полиномиальной.*

Теорема 1.2. *(Гончаров) [6] Существует алгоритм, строящий по любой L_1-формуле φ такую L_0-формулу ψ, что:*

$$HW(\mathfrak{M}) \models \forall \overline{v}\ \varphi(\overline{v}) \Leftrightarrow \psi(\overline{v})$$

По-сути, теорема 1.2 говорит о том, что данное термально-формульное расширение является консервативным. Но при этом логический язык становится более выразительным. Более того, сохраняются и полиномиальные свойства проверки истинности любой L_1-формулы в модели $HW(\mathfrak{M})$.

В *третьем параграфе* определяется понятие р-итерационного терма вида $Iteration_{g,\varphi}$ термальное расширение с помощью которого помогло построить р-полный язык L. В р-итерационном терме базовыми конструкциями выступают L-программа g и L-формула φ, которые также задаются с помощью взаимной индукции. Следующая синтаксическая конструкция задает р-итерационный терм и при этом участвует в индуктивном определении понятия L-программы:

$$Iteration_{g,\varphi}(w,n) = \begin{cases} g^i(w), \text{ если существует} \\ \quad i \le n\ HW(\mathfrak{M}) \models \varphi(g^i(w)) \\ \quad \text{и } \forall j < i\ HW(\mathfrak{M}) \not\models \varphi(g^j(w)) \\ false, \text{ иначе} \end{cases} \tag{1.3}$$

где для L-программы g выполняется условие: $|g(w)| \le |w| + C$.

Теорема 1.3. *В р-вычислимой наследственно-конечной списочной надстройке $HW(\mathfrak{M})$ сигнатуры σ:*
1) Вычислительная сложность любой L-программы является полиномиальной.
2) Вычислительная сложность проверки истинности любой L-формулы является полиномиальной.

Теорема 1.3 доказывается индукцией по сложности L-программ и L-формул. На индукционном шаге рассматривается конкретный вид L-программы $f(t_1,\ldots,t_n)$, $Cond(t_1,\varphi_1,\ldots,t_{n+1})$, $Iteration_{g,\varphi}(t_1,t_2)$ и т.д. Индукционное предположение применяется к L-программам и L-формулам меньшей сложности и отсюда вытекает пункт 1) теоремы 1.3. Для L-формул все аналогично, смотрится их структура, применяется индукционное предположение к L-формулам и L-программам меньшей сложности и отсюда вытекает пункт 2) теоремы 1.3.

Четвертый параграф является своего рода подготовительной базой для доказательства главной теоремы этой главы о равенстве классов P и L.

Дается определение машины Тьюринга, конфигурации машины Тьюринга, определяется операция $\otimes$, которая оказывается p-вычислимой.

В *пятом параграфе* доказывается основная теорема этой главы. Она устанавливает p-полноту языка L.

Теорема 1.4. *(Решение проблемы P=L)*
Пусть $HW(\mathfrak{M})$ – p-вычислимая модель конечной сигнатуры σ тогда:
1) Любая L-программа имеет полиномиальную вычислительную сложность.
2) Для любой полиномиально вычислимой функции существует подходящая L-программа реализующая ее.

Первый пункт автоматически вытекает из теоремы 1.3, второй пункт доказывается с помощью моделирования работы машины Тьюринга реализующей некоторую полиномиально вычислимую функцию f с помощью p-итерационного терма. В силу того, что в построении p-итерационного терма можно указывать количество итераций в виде некоторого многочлена $h(|x|)$, этого достаточно (при правильном выборе многочлена $h(|x|)$), чтобы промоделировать работу машины Тьюринга первые $h(|x|)$ шагов и вычислить значение функции f.

Как следствие, из этой теоремы вытекает тот факт, что в языке L можно реализовать любые алгоритмы полиномиальной вычислительной сложности. Что в свою очередь открывает широкие перспективы применения этого языка в таких направлениях как: искусственный интеллект, робототехника, смарт контракты. Более того, с помощью L-программ можно будет реализовывать алгоритмы в объяснительном искусственном интеллекте, где требуется человекопонятным языком объяснить почему был выдан именно этот результат.

В *шестом параграфе* приводится модифицированная конструкция p-итерационного терма, которая увеличивает выразительную силу языка L, но при этом не выводит за рамки полиномиальности. В модифицированном p-итерационном терме были изменены базовые условия на $g(x)$ и $\varphi(x)$ следующим образом.
Пусть для фиксированного $n \in N$ L-программа $g(x)$ задается следующим образом:

$$g(w) = \begin{cases} w^*, \text{ если } w =< w_1, \ldots, w_n > \\ false, \text{ иначе} \end{cases} \tag{1.22}$$

где $w^* =< w_1^*, \ldots, w_n^* >$ и при этом выполнены следующие условия (с точностью до перестановки):
1) $|w_1^*| \leq |w_1| + C \cdot \sum_{i=2}^{n} |w_i|^p$
2) $|w_i^*| \leq |w_i|$, для любого $i \in [2, \ldots, n]$

Утверждение 1.5. *Терм* $Iteration_{g,\varphi}$ *из* (1.3) *с L-программой* g (1.22), *удовлетворяющей условиям 1) и 2), является p-вычислимым.*

Из утверждения 1.5 и теоремы 1.4 вытекает то, что термальное расширение с помощью так модифицированных p-итерационных термов консервативно.

Вторая глава посвящена решению вопросов полиномиального характера для индуктивно задаваемых множеств и объектов. В этой главе ключевым результатом является построение полиномиального аналога для классической теоремы Ганди.

В *первом параграфе* формулируется классическая теорема Ганди о неподвижной точке. Дается ряд вспомогательных определений связанных с этой теоремой, определяется базовая алгебраическая структура, строится непосредственно сам монотонный оператор.

Теорема 2.1. *(классическая теорема Ганди) [10]*
Пусть $\Phi(P^+)$ *–* Σ*-формула сигнатуры* $\sigma^* \cup < P^k >$, *в которую предикатный символ* P *входит позитивно, и пусть* $\overline{x} = x_0, \ldots, x_{k-1}$ *– список попарно различных переменных такой, что* $FV(\Phi) \subseteq \{\overline{x}\}$. *Тогда наименьшая неподвижная точка* $\Gamma_* \subseteq w^k$ *оператора* $\Gamma^{\Omega^*}_{\Phi[x]}$ *является* Σ*-предикатом на* Ω^*.

Во *втором параграфе* содержатся вспомогательные определения и ряд лемм связанных с расщеплением списка на элементы. Также индуктивно определяется сложность списочного элемента.

В *третьем параграфе* определяется понятие GNF-системы, дается определение p-вычислимой GNF-системы и формулируется ряд замечаний насчет свойств таких систем. P-вычислимые GNF-системы играют основную роль в формулировке и доказательстве полиномиального аналога теоремы Ганди, которое будет приведено в параграфе шесть.

В *четвертом параграфе* доказывается *Лемма о подстановке*. В которой осуществляется оценка сложности подстановки значений $w_1, \ldots, w_k$ вместо свободных переменных в потенциально порождающую L-формулу $\Phi_m(x_1, \ldots, x_k)$.

В *пятом параграфе* определяется непосредственно сам монотонный оператор, обладающий свойством неподвижной точки. Этот оператор строится по заданной p-вычислимой GNF-системе однозначно.

В *шестом параграфе* формулируется и доказывается обобщенный полиномиальный аналог теоремы Ганди о наименьшей неподвижной точке с p-вычислимыми начальными условиями (обобщенная PAG-теорема с p-вычислимыми начальными условиями).

Теорема 2.5. *(обобщенная PAG-теорема с p-вычислимыми начальными условиями)*
Пусть задана p-вычислимая GNF-система G, тогда наименьшая неподвижная точка $\Gamma_w(Q)$ с p-вычислимыми начальными условиями оператора $\Gamma^{HW(\mathfrak{M})}_{F_{P_1},\ldots,F_{P_n}}$ является p-вычислимой.

Доказывается эта теорема индукцией по сложности списочного элемента w. Тонкость доказательства заключается в том, что в индукционном предположении для оценки вычислительной сложности проверки принадлежности элемента w проекции на i-ю координату неподвижной точки $\Gamma_w(Q)$ используется выражение вида $C\cdot(r(w)+1)\cdot|w|^p$, где $r(w)$ – ранг элемента. Как только это удается показать, то в силу того, что $r(w)+1\leq|w|$ можно перейти уже к оценке, что сложность проверки не превосходит $C\cdot|w|^{p+1}$.

PAG-теорема является одним из основных инструментов, наряду с p-итерационными термами, для доказательства полиномиальной вычислимости индуктивных множеств различной природы. В том числе с помощью нее были доказаны большинство теорем из главы 4, а также доказана теорема 3.14 для множества выводов в грамматике G из главы 3.

Следующее следствие из этой теоремы дает оценку на степень полинома, ограничивающего вычислительную сложность наименьшей неподвижной точки.

Следствие 2.5.1. *Пусть в p-вычислимой GNF-системе с полиномиально вычислимыми начальными условиями вычислительная сложность всех базовых функций, предикатов, а также функций γ_i для построения порождающих формул не превосходит $O(|x|^p)$. Тогда вычислительная сложность наименьшей неподвижной точки не превосходит $O(|x|^{p+2})$.*

Третья глава посвящена вопросам существования полиномиально вычислимых списочный представлений для различных множеств объектов математической логики и теории программирования. Исследуются вопросы существования таких представления для множества формул, термов и множества доказательств логики предикатов первого порядка, для множества L-программ и L-формул. Также было показано существование p-вычислимого списочного представления для множества выводов в порождающих грамматиках.

В *первом параграфе* строятся p-вычислимые GNF-системы для множества списочных представлений термов (предикат $Term$) и формул (предикат $Formula$) логики предикатов первого порядка [11]. После этого применяется PAG-теорема и, как следствие, вытекают следующие леммы.

Лемма 3.1. *$Term$ – p-вычислимый предикат.*

Лемма 3.2. *$Formula$ – p-вычислимый предикат.*

Во *втором параграфе* аналогичным способом доказывается существование полиномиально вычислимых списочных представлений для L-формул и L-программ. Здесь также строятся подходящие p-вычислимые GNF-системы, в основе которых лежат порождающие семейства для расширяемых предикатов $Formula_L$ и $Program_L$.

Лемма 3.5. *$Formula_L$ и $Program_L$ – p-вычислимые предикаты.*

Третий параграф содержит ряд вспомогательных лемм, которые потребуются для доказательства основной теоремы этой главы. Здесь определяется p-вычислимый предикат *checkAddmissible*, а также p-вычислимые функции *changeFreeVar*, *change*, *gtp*, *gt*, *ChangeWithCond*.

Четвертый параграф посвящен списочным представлениям для секвенций и аксиом, которые нам потребуются для доказательства существования полиномиально вычислимого списочного представления для множества доказательств в исчислении предикатов первого порядка. Вводятся предикаты *Sec* и *Axiom* и показывается, что эти предикаты p-вычислимы.

Пятый параграф посвящен доказательству основной теоремы этого раздела. В данном параграфе явным образом строится порождающее семейство для предиката *Proof* являющегося списочным представлением для множества доказательств ИП в виде дерева. В силу того, что удается построить p-вычислимую GNF-систему, то отсюда вытекает следующая важная теорема.

Теорема 3.12. *Proof – p-вычислимый предикат.*

Эта теорема позволяет гарантировать существование алгоритмов полиномиальной вычислительной сложности для проверки доказательства секвенции в виде дерева. И может быть использована в направлении объяснительного искусственного интеллекта для проверки корректности логических выводов.

В *шестом параграфе* доказывается утверждение, что множество линейных доказательств в ИП также имеет полиномиально вычислимое списочное представление.

Следствие 3.12.1. *lProof – p-вычислимый предикат.*

Доказательство этого утверждения получается с помощью построения специальной L-программы, в которой в качестве базовых формул используются видоизмененные формулы порождающего семейства для предиката *Proof*.

Теорема 3.12 и следствие 3.12.1 имеют важное практическое значение в направлении искусственного интеллекта. Где важно проверить корректность рассуждений интеллектуальной машины или робота.

Седьмой параграф посвящен порождающим грамматикам. Здесь основным результатом является существование полиномиально вычислимого списочного представления для множества выводов в порождающей грамматике G. Для этих целей формируется два множества. Обозначим за $L(G)_{\vdash}^{V^*}$ множество выводов из аксиомы S цепочек $x \in V^*$ закодированных в виде списков следующего вида:

$$L(G)_{\vdash}^{V^*} = \{<<< \cdots << S >, \alpha_1 >, \cdots >, \alpha_{n_x-1} >, x >: \\ S \vdash \alpha_1 \vdash \cdots \vdash \alpha_{n_x-1} \vdash x\} \tag{3.18}$$

где V - множество символов терминального алфавита грамматики G.

Обозначим за $L(G)_{\vdash}^{D^*}$ множество выводов из аксиомы S цепочек вида $x \in D^*$:

$$L(G)_{\vdash}^{D^*} = \{<<< \cdots << S >, \alpha_1 >, \cdots >, \alpha_{n_x-1} >, x >: \\ S \vdash \alpha_1 \vdash \cdots \vdash \alpha_{n_x-1} \vdash x\} \tag{3.19}$$

где D - множество терминальных и нетерминальных символов грамматики G.

Теорема 3.14. *$L(G)_{\vdash}^{D^*}$ из (3.19) является р-вычислимым множеством.*

Теорема 3.14 доказывается с помощью PAG-теоремы. Для этого строится р-вычислимая GNF-система с подходящими свойствами.

Теорема 3.15. *$L(G)_{\vdash}^{V^*}$ из (3.18) является р-вычислимым множеством.*

Теорема 3.15 доказывается с помощью построения подходящей L-программы, где множество $L(G)_{\vdash}^{D^*}$ выступает в качестве одноместного р-вычислимого предиката.

В качестве применения тех результатов, которые были получены в первых трех главах, в **четвертой главе** на базе р-полного языка L строится высокоуровневый объектно-ориентированный язык L*. Данный язык является консервативным расширением языка L, при этом синтаксис языка очень схож с синтаксисом таких высокоуровневых языков как C++, PHP, JavaScript и т.д. Что в свою очередь позволяет использовать его в таких важных направлениях как: искусственный интеллект (в том числе объяснительный), робототехника, смарт-контракты, машинное обучение и т.д. Более того, в этом языке можно реализовывать любые алгоритмы машинного обучения, в том числе алгоритмы реализованные с помощью нейронных сетей прямого распространения.

В *первом параграфе* дается определение виртуальной машины V, которая является тройкой следующего вида:

$$V = (HW(\mathfrak{M}), \sigma, < Fl, Cl, Hl, Sl, Ll >)$$

где Fl будем называть списком L*-определимых функций, Cl – списком L*-определимых классов, Hl – вспомогательный список, Sl – список, выполняющий роль экрана для виртуальной машины V, Ll – список, выполняющий роль ленты для виртуальной машины V. Сама же $HW(\mathfrak{M})$ сигнатуры σ является p-вычислимой моделью.

И, далее, с помощью взаимной индукции определяются такие понятия как L(V)-программы, L(V)-формулы, а также L*-программы.

Второй и *третий параграфы* посвящены описанию синтаксических конструкций, задающих соответственно функции и классы в новом языке L*. А также определяются такие понятия как наследование, свойства и методы класса, инициализация объекта класса и т.д.

В *четвертом параграфе* вводится понятие функции компиляции:

$$C :<< Fl,Cl > ,p > \rightarrow << Fl^*,Cl^* > ,p^* >$$

которая переводит L*-программу p в L*-программу p^* параллельно изменяя списки Fl и Cl. Эта функция удаляет синтаксические конструкции задающие функции и классы из p и записывает их в виде списочных представлений в Fl и Cl соответственно. На выходе получается L*-программа p^* без синтаксических конструкций описывающих функции и классы.

В качестве замечания указывается, что функция компиляции C является p-вычислимой.

В *пятом параграфе* описывается процесс исполнения L*-программ виртуальной машиной V. Причем за один шаг машина V сразу вычисляет значение любой L-программы и производит проверку истинности любой L-формулы в p-вычислимой модели $HW(\mathfrak{M})$. Заметим, что L-программы и L-формулы не содержат нестандартные L(V)-конструкции. Поэтому можно считать, что $HW(\mathfrak{M})$ сигнатуры σ с L-программами и L-формулами является оракулом для V при ее работе с L*-программой.

Шестой параграф посвящен доказательству консервативности данного расширения. Для того чтобы доказать консервативность, достаточно показать, что любая L*-программа p имеет полиномиальную сложность. Для этого достаточно смоделировать процесс исполнения машиной V программы p^* с помощью модифицированного p-итерационного терма в подходящей наследственно-конечной списочной надстройке $HW(M)$ конечной сигнатуры σ. В которой любая L*-программа p принадлежала бы множеству M.

Теорема 4.3. *(о p-вычислимости L*-программ)*
Любая L-программа является p-вычислимой.*

Теорема 4.4. *(о консервативности расширения)*
Язык L является консервативным расширением языка L.*

Заключение

В диссертации решены следующие проблемы:

1. Совместно с Гончаровым автором был разработан p-полный логический язык программирования L и тем самым была решена проблема равенства классов P и L.
2. Совместно с Гончаровым автором был доказан полиномиальный аналог классической теоремы Ганди о наименьшей неподвижной точке.
3. Автором было показано существование полиномиально вычислимых представлений для базовых конструкций ИП: множества термов ИП, множества формул ИП, а также для множества доказательств в ИП (как линейных, так и в виде дерева).
4. Автором было показано существование полиномиально вычислимых представления для L-формул и L-программ.
5. Автором также было показано существование полиномиально вычислимых представлений для множества выводов в порождающих грамматиках.
6. Совместно с Гончаровым, для выполнения поставленных задач на основе GNF-систем, PAG-теоремы и L-программ, автором был создан метод и разработаны техники, которые позволяют исследовать полиномиальную сложность с точки зрения формульной определимости и построения подходящих L-программ.
7. Совместно с Гончаровым на базе логического языка L был разработан p-полный объектно-ориентированный язык L^* синтаксис которого очень схож с синтаксисами других высокоуровневых объектно-ориентированных языков программирования.

Построение p-полного языка L дало толчок к использованию данных теоретических наработок в практических целях. Это касается таких направлений как: объяснительный искусственный интеллект, умные города [40], смарт контракты, робототехника и т.д. В любых этих направлениях требуется описать работу того или иного процесса. И для этих целей как нельзя лучше подходит построенный нами логический язык программирования L. Помимо практического применения автора интересуют и теоретические продолжения исследований. Это касается более полной формализации понятия блокчейна, алгоритмов консенсуса и теоретическое описание различных устойчивых децентрализованных систем. Все это планируется реализовать в рамках концепции семантического программирования.

Благодарности

В заключение хочу выразить свою благодарность и большую признательность своему научному руководителю Гончарову С. С. за поддержку, помощь, обсуждение результатов и научное руководство. А также за тот неоценимый научный опыт, который он передал мне в процессе нашей совместной работы. Также хочу поблагодарить Алаева П. Е. за консультации, корректировки и анализ ряда моих результатов. Выражаю благодарность Одинцову С. П. за активную поддержку моих исследований и становления меня как ученого. Выражаю особую признательность Бутурлакину А. А. за помощь и консультации связанные с вопросами алгебры. Хочу поблагодарить весь коллектив кафедры алгебры и логики, кто сделал данную работу возможной. Также я благодарен своей жене Нечесовой Н. С. за неизменную поддержку и своей маме Нечесовой Е. И. за веру в меня.

Литература

[1] *Алаев П.Е.* Структуры, вычислимые за полиномиальное время. I. *Алгебра и логика.* **2016**. *55(6)*, стр.647–669. https://doi.org/10.17377/alglog.2016.55.601

[2] *Алаев П.Е.* Структуры, вычислимые за полиномиальное время. II. *Алгебра и логика.* **2017**. *56(6)*, стр.651–670. https://doi.org/10.17377/alglog.2017.56.601

[3] *Алаев П.Е.* Существование и единственность структур, вычислимых за полиномиальное время. *Алгебра и логика.* **2016**. *55(1)*, стр.106–112. https://doi.org/10.17377/alglog.2016.55.107

[4] *Алаев П.Е., Селиванов В.Л.* Поля алгебраических чисел, вычислимые за полиномиальное время. I *Алгебра и логика.* **2019**. *58(6)*, стр.673–705. https://doi.org/10.33048/alglog.2019.58.601

[5] *Анцыферов С.С.* Проблемы искусственного интеллекта. *Проблемы искусственного интеллекта.* **2015**. *0(1)*, стр.5-12

[6] *Гончаров С.С.* Условные термы в семантическом программировании. *Сибирский математический журнал.* **2017**. *58(5)*, стр.794–800. https://doi.org/10.17377/smzh.2017.58.506

[7] *Гончаров С.С., Свириденко Д.И.* Логический язык описания полиномиальной вычислимости. *Доклады Академии Наук.* **2019**. *485(1)*, 11–14. https://doi.org/10.31857/S0869-5652485111-14

[8] *Гончаров С.С., Свириденко Д.И.* Σ-программирование. *Вычислимые системы.* **1985**. *485(1)*, 3–29.

[9] *Гончаров С.С., Свириденко Д.И.* Семантическое моделирование и искусственный интеллект. *Сибирский философский журнал.* **2018**. *16(4)*, стр.5-25. https://doi.org/10.25205/2541-7517-2018-16-4-5-25

[10] *Ершов Ю.Л.* Определимость и вычислимость. *Новосибирск.* **2000**

[11] *Ершов Ю.Л, Палютин Е.А.* Математическая логика. *Наука.* **1987**

[12] *Малых А.А., Манцивода А.В.* Документное моделирование. *Изв. ИГУ. Серия: Математика.* **2017**. *21*, стр.89-107. https://doi.org/10.26516/1997-7670.2017.21.89

[13] *Малых А.А., Манцивода А.В.* Libretto: язык программирования как средство логического объектного моделирования. *Сборник докладов международной конференции «Мальцевские чтения»*, Новосибирск, **2011**. стр.130-131

[14] *Манцивода А.В., Пономарев Д.К.* Формализация документных моделей средствами семантического моделирования *Изв. ИГУ. Серия: Математика.* **2019**. *27*, стр.36–54. https://doi.org/10.26516/1997-7670.2019.27.36.

[15] *Пратт Т., Зелковиц М.* Языки программирования: разработка и реализация. *4-е изд. СПб.: Питер.* **2003**

[16] *Тестелец Я. Г.* Введение в общий синтаксис. Глава XI. Порождающая грамматика: от правил к ограничениям. *РГГУ.* **2001**. ISBN 5-7281-0343-X.

[17] *Barwise J.* Admissible Sets and Structures. *Springer.* **1975**

[18] *Bathaee Y.* The artificial intelligence black box and the failure of intent and causation. *Harvard Journal of Law & Technology.* **2018**. *31(2)*

[19] *Cenzer D., Remmel J.* Polynomial-time versus recursive models. *Ann. Pure Appl. Log..* **1991**. *54*, 17–58.

[20] d0sl – programming language. URL: https://d0sl.org/

[21] *Deisenroth, M.P.; Faisal, A.A.; Ong. C.S.* Mathematics for Machine Learning *Published by Cambridge University Press*, **2020**

[22] *Ershov Y.L., Goncharov S.S., Sviridenko D.I.* Semantic programming. *In Proceedings of the Information Processing 86* **1986**. *10*, p.1113–1120

[23] *Ershov Y.L., Goncharov S.S., Sviridenko D.I.* Semantic foundations of programming. *Lecture Notes in Computer Science.* **1987**. *278*, p.116–122.

[24] *Goncharov S.S., Sviridenko D.I.* Recursive terms in semantic programming. *Sib. Math. J.* **2018**. *59*, p.1014–1023. https://doi.org/10.1134/S0037446618060058

[25] *Goncharov S.S., Sviridenko D.I.* Theoretical aspects of Σ-programming. *Lecture Notes in Computer Science.* **1986**. *215*, p.169–179.

[26] *Goncharov S.S., Ospichev S.S., Ponomaryov D.K., Sviridenko D.I.* The expressiveness of looping terms in the semantic programming. *Sib. Electron. Math. Rep.* **2020**. *17*, p.380–394. https://doi.org/10.33048/semi.2020.17.024

[27] *Gumirov V., Matyukov P., Palchunov D.* Semantic Domain-specific Languages. *SIBIRCON.* **2019**. https://doi.org/10.1109/SIBIRCON48586.2019.8958237

[28] *Hagan, M.T; Demuth, H.B.; Beale, M.H; Jesus, O.D.* Neural Network Design *Martin Hagan*, **2014**

[29] *Lewis H., Papadimitriou C.* Elements of the Theory of Computation. *Prentice-Hall: Upper Saddle River, NJ, USA.* **1998**

[30] *Michaelson, G.* Programming Paradigms, Turing Completeness and Computational Thinking. *The Art, Science, and Engineering of Programming.* **2020**. *4(3).* https://doi.org/10.22152/programming-journal.org/2020/4/4

[31] *Ospichev S.S.; Ponomaryov D.K.* On the complexity of formulas in semantic programming. *Sib. Electron. Math. Rep.* **2018**. *15*, p.987–995. https://doi.org/10.17377/semi.2018.15.083

[32] *Papadimitriou C.* Computational complexity. *Addison-Wesley.* **1994**

[33] *Russel, S.J; Norvig, P.* Artificial Intelligence - A Modern Approach (3rd Edition) *Prentice Hall*, **2010**

[34] *Wielemaker, J; Hildebrand, M; Ossenbruggen, J.* Using Prolog as the Fundament for Applications on the Semantic Web. *Proceedings of the ICLP2007 Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services (ALPSWS2007)*, **2007**. p.1–16, RWTH Aachen.

Работы автора по теме диссертации из журналов списка ВАК

[35] *Goncharov, S.S.; Nechesov, A.V.* Polynomial analogue of Gandy's fixed point theorem. *Mathematics* **2021**. *9(17):2102.* https://doi.org/10.3390/math9172102

[36] *Goncharov S.S., Nechesov A.V.* Solution of the problem P = L. *Mathematics.* **2022**. *10(1):113.* https://doi.org/10.3390/math10010113

[37] *Нечесов, А.В.* Некоторые вопросы полиномиально вычислимых представлений для порождающих грамматик и форм Бэкуса-Наура. *Математические труды.* **2022**. *25(1)*, стр.134-151. https://doi.org/10.33048/mattrudy.2022.25.106

[38] *Нечесов, А.В.* Семантическое программирование и полиномиально вычислимые представления. *Математические труды.* **2022**. *25(2)*, стр.174-202. https://doi.org/10.33048/mattrudy.2022.25.208

[39] *Nechesov, A.V.* Some Questions on Polynomially Computable Representations for Generating Grammars and Backus–Naur Forms. *Sib. Adv. Math.* **2022**. *32*, p.299–309. https://doi.org/10.1134/S1055134422040058

Работы автора по теме диссертации из журналов не из списка ВАК

[40] *Nechesov A.V.; Safarov R.A.* Web 3.0 and smart cities. *Сборник докладов республиканской научно-технической конференции «Современное состояние и перспективы развития цифровых технологий и искусственного интеллекта». Самарканд, Узбекистан*, **2022**. ч.1, стр.248-253

[41] *Гончаров С.С., Нечесов А.В.* Объектно-ориентированный логический язык программирования для искусственного интеллекта и робототехники. *Proceedings of the International Conference «Mathematical Logic and Computer Science», ENU, Астана, Казахстан*, **2022**. стр.191-195

Нечесов Андрей Витальевич

Полиномиальная вычислимость
в семантическом программировании

Автореф. дис. на соискание ученой степени канд. физ.-мат. наук

Подписано в печать ___.___._____. Заказ № _____
Формат 60×90/16. Усл. печ. л. 1. Тираж 100 экз.
Типография _____