

Московский Государственный Университет
имени М.В. Ломоносова
Российская Академия Наук
Международная Академия Технологических Наук

Интеллектуальные Системы.

Теория и приложения

ТОМ 27 ВЫПУСК 4 * 2023

МОСКВА

Главный редактор: д.ф.-м.н., профессор Э.Э. Гасанов

Редакционная коллегия:

д.ф.-м.н., проф. А. Е. Андреев	(зам. главного редактора)
к.ф.-м.н., с.н.с. А.В. Галащенко	(зам. главного редактора)
к.ф.-м.н., доц. А. С. Строгалов	(зам. главного редактора)
к.ф.-м.н., м.н.с. В. В. Осокин	(ответственный секретарь)

д.ф.-м.н, проф. В.В.Александров, д.ф.-м.н, проф. С.В.Алешин, д.ф.-м.н, проф. Д.Н.Бабин, проф. К.Вашик, проф. Я.Деметровиц, академик РАН, д.ф.-м.н, проф. Ю.Л.Ершов, проф. Г.Килибарда, д.ф.-м.н, проф. В.Н.Козлов, д.ф.-м.н, проф. А.В.Михалев, к.ф.-м.н, в.н.с. В.А.Носов, д.ф.-м.н, проф. А.С.Подколзин, д.ф.-м.н, проф. Ю.П.Пытьев, д.т.н, проф. А.П.Рыжов, академик РАН, д.т.н, проф. А.С.Сигов, проф. Б.Тальхайм, проф. Ш.Ушчумлич, д.ф.-м.н, доц. А.А.Часовских, д.ф.-м.н, проф. А.В.Чечкин, к.ф.-м.н. Р.Шчепанович.

Секретарь редакции: И. О. Бергер, Е. В. Кузнецова

В журнале «Интеллектуальные системы. Теория и приложения» публикуются научные достижения в области теории и приложений интеллектуальных систем, новых информационных технологий и компьютерных наук.

Издание журнала осуществляется под эгидой МГУ имени М. В. Ломоносова, Научного Совета по комплексной проблеме «Кибернетика» РАН, Отделения «Математическое моделирование технологических процессов» МАТН.

Учредитель журнала: ООО «Интеллектуальные системы».

Журнал входит в список изданий, включенных ВАК РФ в реестр публикаций материалов по кандидатским и докторским диссертациям по математике и механике.

Спонсором издания является:

ООО «Два Облака»

Разработка корпоративных информационных систем

<http://www.dvaoblaka.ru>

Индекс подписки на журнал: 64559 в каталоге НТИ «Роспечать».

Адрес редакции: 119991, Москва, ГСП-1, Ленинские Горы, д. 1, механико-математический факультет, комн. 12-01.

Адрес издателя: 115230, Россия, Москва, Хлебозаводский проезд, д. 7, стр. 9, офис 9. Тел. +7 (495) 939-46-37, e-mail: mail@intsysjournal.org

*) Прежнее название журнала: «Интеллектуальные системы».

© ООО «Интеллектуальные системы», 2023.

ОГЛАВЛЕНИЕ

Часть 1. Общие проблемы теории интеллектуальных систем

Часовских А.А., Половников В.С., Хусаенов А.А., Боков Г.В., Коновалов А.Ю., Дробышев А.С., Бирюкова В.А. Автоматизация поиска архитектур искусственных нейронных сетей 5

Часть 2. Специальные вопросы теории интеллектуальных систем

Дергач П.С., Бахрамова Д.Б. Об автоматных неисправностях при алфавитном кодировании 29

Корчагин Н.П. Сложность задачи о существовании сюръективного гомоморфизма на рефлексивные циклы 40

Часть 3. Математические модели

Алексиадис Н.Ф. Замкнутые классы в функциональной системе рациональных функций с рациональными коэффициентами 63

Бабин Д.Н., Летуновский А.А. О выразимости автоматов с операцией суперпозиции 76

Дергач П.С., Ботирова Н.С. Об изменении длины минимальной склейки при алфавитных неисправностях 79

Часть 1.
Общие проблемы теории
интеллектуальных систем

Автоматизация поиска архитектур искусственных нейронных сетей

А. А. Часовских¹, В. С. Половников², А. А. Хусаенов³,
Г. В. Боков⁴, А. Ю. Коновалов⁵, А. С. Дробышев⁶,
В. А. Бирюкова⁷

Представлен обзор работ о методах автоматизации генерирования архитектур искусственных нейронных сетей. Наиболее подробно излагается материал по использованию для этого больших

¹ *Часовских Анатолий Александрович* — доктор физико-математических наук, профессор каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: chasovskikh@mail.ru.

Chasovskikh Anatoly Alexandrovich — Doctor of Physical and Mathematical Sciences, Professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

² *Половников Владимир Сергеевич* — кандидат физико-математических наук, старший научный сотрудник каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: pvser@mail.ru

Polovnikov Vladimir Sergeevich — Candidate of Physical and Mathematical Sciences, Senior Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

³ *Хусаенов Артём Азатович* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: a.khusaenov@mail.ru

Khusaenov Artem Azatovich — Postgraduate, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

⁴ *Боков Григорий Владимирович* — кандидат физико-математических наук, доцент каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: bokovgrigoriy@gmail.com

Bokov Grigoriy Vladimirovich — Candidate of Physical and Mathematical Sciences, Associate Professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

⁵ *Коновалов Александр Юрьевич* — кандидат физико-математических наук, младший научный сотрудник лаб. математических проблем искусственного интеллекта мех.-мат. ф-та МГУ, e-mail: alexandr.konoval@gmail.com

Konovalov Aleksandr Yurievich — Candidate of Physical and Mathematical Sciences, Junior Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Laboratory of Mathematical Problems of Artificial Intelligence.

⁶ *Дробышев Александр Сергеевич* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: drobyshev.sanya@yandex.ru

Drobyshev Alexander Sergeevich — Postgraduate, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

⁷ *Бирюкова Вероника Андреевна* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: biryukovaveronika@mail.ru

Biryukova Veronika Andreevna — Postgraduate, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

языковых моделей. Для экспериментов использован ChatGPT & Midjourney, который доступен в мессенджере Telegram.

Ключевые слова: искусственная нейронная сеть, большая языковая модель, поиск архитектуры нейронной сети, генеративный предварительно обученный преобразователь.

1. Введение.

Поиск оптимальной архитектуры искусственной нейронной сети (ИНС) – важная задача глубокого обучения. Процедуры автоматизации этого процесса объединены общим термином Neural Architecture Search (NAS). Методика решения заключается в определении целевой функции, позволяющей оптимизировать точность функционирования нейронной сети путем варьирования ее характеристик при ограничении на количество итераций варьирования. Это типичный подход, при выполнении которого тестируются и отбраковываются тысячи архитектур нейронных сетей. Каждая итерация заключается в обучении и тестировании модели ИНС на выбранной базе данных.

Разработка хорошей обучаемой архитектуры ИНС – сложный процесс для экспертной разработки. NAS предполагает автоматическое нахождение архитектуры нейронной сети, которая будет иметь высокую эффективность после обучения. В обзоре [1] по NAS, вышедшем 2019 году, дана классификация разработанных к тому времени систем в трех измерениях: пространство поиска архитектур ИНС, стратегия поиска и стратегия оценки производительности.

Пространство архитектур, в котором осуществляется поиск нужной архитектуры ИНС, может быть существенно ограничено, если используются экспертные знания о характере решаемой задачи и типичных свойствах архитектур. С другой стороны, такие ограничения могут исключить эффективные не изученные варианты архитектур. Наиболее простые пространства архитектур – цепи, состоящие из слоев соединенных последовательно. Более сложный вариант – ИНС с ветвящейся архитектурой. Работа [2] посвящена стратегии построения архитектур из блоков. Оптимизация отдельных блоков, выбор их количества и порядка соединения позволяет существенно сократить время поиска нужной архитектуры. Построение сетей из однородных блоков продемонстрировало эффективность в LSTM и RNN. При таком подходе к поиску архитектур возникает микроуровень для оптимизации ячеек и макроуровень для построения сети из ячеек.

Стратегия поиска в пространстве архитектур, которое может быть бесконечным, реализует компромисс между временем, затрачиваемым на поиск и эффективностью полученного решения. К основным стра-

тегиям поиска в пространстве архитектур, согласно [1], следует отнести случайный поиск, байесовскую оптимизацию, эволюционные методы, обучение с подкреплением (Reinforcement Learning, RL) и градиентные методы. Использование NAS с байесовской оптимизацией позволило получить современные архитектуры машинного зрения [3] и ИНС, способные превзойти результаты распознавания экспертов [4]. Низкая сложность байесовских методов объясняет их широкую популярность [5]. Но используются они чаще для оптимизации гиперпараметров сетей, как более эффективные для оптимизации в пространствах низкой размерности.

Метод обучения с подкреплением, связан с обучением агента (как правило, другой нейронной сети) предлагать архитектуры нейронных сетей для решения данной задачи, используя показатель эффективности предложенной архитектуры после ее обучения в качестве вознаграждения [6]. Большинство известных алгоритмических подходов используют обучение с подкреплением или эволюционные алгоритмы.

Можно считать, что эволюционный подход продолжает исследования по нейроэволюции, начатые еще в 1970-х годах. При использовании эволюционных методов нужную архитектуру ищут непосредственно в пространстве архитектур ИНС, без использования генератора [7], а эффективность обученной ИНС служит показателем приспособляемости. К недостаткам эволюционных алгоритмов следует отнести то, что они не используют большого накопленного опыта генерирования ИНС. Этим может объясняться большое количество перебираемых архитектур ИНС в процессе их работы.

Метод случайного поиска по дереву (Monte Carlo Tree Search) тоже использует поиск в пространстве возможных архитектур [8]. Во всех перечисленных случаях от эксперта требуется определение пространства допустимых архитектур.

Стратегия оценки производительности связана с сокращением времени на проверку эффективности разрабатываемого решения. Самый простой способ оценивания – обучить выбранную модель ИНС и протестировать ее, но при многократном выполнении это приводит к высоким затратам по времени. Следует выделить следующие подходы для сокращения общего времени обучения:

- сокращение количества эпох обучения, обучение на подмножестве данных, использование уменьшенных архитектур;
- экстраполяцию показателей эффективности после нескольких эпох обучения;
- использование весов предыдущей обученной модели для определения стартовых весов новой модели;

- обучение единственной модели, веса которой распределяются между различными архитектурами, которые являются подграфами, вообще говоря, пересекающимися, обученной модели.

Но наибольшая производительность при оценке эффективности предложенной архитектуры достигается, если использовать предсказание качества этой архитектуры, например, другой нейронной сетью [9].

Для того, чтобы не ограничивать проектировщика в выборе архитектур путем определения пространства архитектур, целесообразно использовать большие языковые модели (Large Language Models, LLM), которые обучаются на больших массивах данных, например, на открытых репозиториях из Github. Эффективность при генерации программного кода, большой объем обучающего материала по ранее разработанным эффективным архитектурам ИИС является важным аргументом в пользу использования LLM.

2. Большие языковые модели.

Широкому использованию LLM способствовало открытие трансформеров [10]. Большинство предшествовавших трансформерам системы обработки последовательной информации было разработано на основе сложных рекуррентных или сверточных архитектурах, которые включали кодировщик и декодер. Наиболее эффективные из этих моделей связывали кодировщик и декодер через механизм внимания. Трансформер, предложенный в работе [10], имел простую архитектуру кодировщик-декодер без рекуррентных и сверточных включений, основанную исключительно на механизме внимания в обеих частях архитектуры (кодировщик, декодер). Эксперименты на двух задачах перевода, с английского языка на немецкий и с английского на французский, показали лучшие результаты. При этом трансформеры поддаются хорошо распараллеливаются и требуют меньшего времени на обучение, чем предшествующие модели.

Открытие трансформеров [11]–[13] способствовало взрывному росту больших языковых моделей. Ориентированный на конкретную прикладную задачу подход, используемый для решения поставленной задачи, связанной с обработкой естественного языка (система вопрос-ответ, автоматический перевод, реферирование и др.), основан на процедуре обучения с учителем на специально разработанной обучающей базе данных. В [11] для решения языковых задач предложено использовать генеративного предварительно обученного преобразователя (generative pre-training transformer, GPT). Процесс обучения модели GPT можно выполнить в два этапа. На первом этапе выполняется обучение без учителя, на втором этапе выполняется тонкая настройка с учителем. Авторами показано,

что прикладные задачи могут быть решены с использованием предварительно обученной языковой модели GPT на неразмеченном большом корпусе языковых данных методом обучения с подкреплением при ее последующей дискриминационной тонкой настройке (discriminative fine-tuning) на данных, отражающих специфику прикладной задачи. Эффективность разработанного GPT по сравнению с существовавшими на тот момент решениями продемонстрирована на 9 из 12 изученных авторами языковых задач. В работе [12] предложена сеть GPT-2, содержащая 1,5 миллиарда параметров, обученная на большой языковой базе WebText, которая справляется с некоторыми прикладными задачами без дополнительного обучения на специально разработанных для этих задач обучающих языковых базах данных.

Использование трансформеров позволило не только создавать современные модели для основных задач обработки естественного языка, они привели к созданию моделей для широкого спектра других задач, таких как разработка кода видеоигр [14], [15]. EvoProming [16] – метод, использующий код LLM в качестве операторов мутации и скрещивания для реализации NAS в задачах классификации MNIST-1D [17] и тестирования алгоритмических рассуждений CLRS [18]. Поскольку эффективность модели можно повысить тривиальным образом, увеличивая количество параметров, большие модели дополнительно штрафуются из-за своего размера. Таким образом, предпочтение отдается малым моделям.

3. Поиск архитектуры ИНС с использованием метода LLMatic.

В работе [19] представлен метод автоматического поиска архитектур ИНС с использованием LLM и разработанного авторами генетического алгоритма. Метод использует характеристику сложности модели как показатель разнообразия, поиск высокоэффективных моделей выполняется при вариации их размера. Следует отметить, что LLM не имеет механизмов оптимизации решений. Для этого авторы работы [19] предлагают использовать генетический Quality-Diversity (QD) алгоритм. QD-метод [20] объединяет семейство эволюционных алгоритмов, которые в дополнение к оптимизации метрики приспособляемости находят разнообразные решения в соответствии с индивидуальными особенностями использования модели. Вместо того, чтобы сохранять популяцию наиболее эффективных особей, QD-методы, такие, как рассмотренный в [21] MAP-Elites, сохраняют популяции для заданных диапазонов «поведенческих характеристик».

QD-методы могут использоваться для управления роботами. Например, если оптимальная траектория окажется недоступной из-за препятствия или поломки, придётся использовать другие решения. Эти методы способствуют устойчивости функционирования решений, избеганию попадания популяций в процессе эволюции в локальный минимум.

В работе [19] предложено соединить возможности генерации кода LLM с разнообразием и устойчивостью генетического QD – алгоритма. Это позволило авторам разработать алгоритм LLMatic для автоматизации нахождения архитектуры ИНС. Если в других работах LLM управляет NAS напрямую через подсказки, то LLMatic использует QD-алгоритм для формирования разнообразных и надежных решений.

Алгоритм LLMatic протестирован авторами на базе MNIST. Показана возможность построения эффективной архитектуры за примерно 2000 поисковых запросов без использования предварительной информации о базе данных и без использования какой-либо известной заранее эффективно работающей на этой базе модели. При этом интерес для авторов представляет поиск не единственного решения, оптимизирующего заданную целевую функцию, а определение набора решений для использования в различных условиях. Например, различие в оптимальных архитектурах может возникать из-за разного объема оперативной памяти на клиентском устройстве. LLMatic начинает поиск с очень простой нейронной сети, разработанной в соответствии с работой Стенли [22], в которой предполагается, что нейроэволюция происходит успешнее, если начинается с простой сети. Эта сеть содержит всего один сверточный слой и один полносвязный слой. Для LLMatic используются метод двунаправленной совместной оптимизации, в котором два отдельных архива используются для хранения взаимодополняющих компонентов, которые можно объединить для решения задачи поиска наиболее «приспособляемой» популяции сетей. В первом архиве сохраняется нейронная сеть, для которой характеристиками поведения являются значения глубины и ширины, а также количество операций с плавающей точкой в секунду (FLOPS). FLOPS был выбран вместо количества параметров, поскольку FLOPS лучше коррелирует с фактическим временем, затраченным на обучение сети. Этот архив назван «архив сетей». Значением приспособляемости для этой сети является значение ошибки на тестовом множестве, достигнутой после обучения сети. Второй архив называется «архив подсказок». Он содержит подсказку и температуру для генерации кода, которые также являются поведенческими дескрипторами. Выбор подсказки и температуры зависит от того, добавлена ли сгенерированная сеть в архив сетей и насколько она обучаемая. Значение приспособляемости для подсказки зависит также от того, насколько добавленная сеть эффективнее сети предыдущего поколения.

Внешний цикл алгоритма LLMatic индексируется номером поколения. Количество поколений фиксировано и является параметром алгоритма. В течении одного поколения осуществляется проверка заполнения обоих архивов. Если архивы недостаточно заполнены, то заданное количество раз с использованием случайно выбранной подсказки, начального значения температуры, значения ошибки наилучшей из сгенерированных сетей операцией мутации исходной сети генерируется новая сеть, температура и новая подсказка, которой добавляются в соответствующие архивы. После заполнения обоих архивов до заданных объемов, в каждом поколении алгоритм LLMatic в течении заданного количества раундов выполняет операции мутации или скрещивания. На каждом раунде выполняется генерация новой сети, а во время операции мутации, кроме того, генерируется новая подсказка. По прошествии очередного поколения в результате тестирования новых сгенерированных сетей обновляется минимальная ошибка сети, и архивы пополняются сгенерированными сетями и подсказками. Архив сетей и архив подсказок используются для выполнения операций мутации и скрещивания.

4. Использование метода GPT-NAS

Идея метода автоматического генерирования моделей GPT-NAS состоит в том, что генерирующая модель, предварительно обученная на корпусе данных большого объема, может обобщить фундаментальные принципы построения нейронных архитектур. Таким образом, GPT-NAS использует модель GPT чтобы сформировать подходящие компоненты архитектуры с использованием базовых, а затем использует эволюционную стратегию для поиска оптимального решения. Такой подход может значительно сократить пространство поиска за счет введения в процесс поиска предварительных знаний. Обширные экспериментальные результаты, выполненные авторами работы [23], показывают, что результаты использования метод GPT-NAS значительно превосходят семь нейронных архитектур, разработанных вручную, и тринадцать архитектур, предоставляемых конкурирующими методами NAS. Кроме того, эти эксперименты также показывают, что предложенный алгоритм повышает производительность точной настройки нейронных архитектур примерно на 12 процентов по сравнению с алгоритмами, не использующими GPT, что дополнительно демонстрирует его эффективность при поиске нейронных архитектур.

GPT-NAS разработан с целью оптимизации поиска оптимальной архитектуры ИНС и пространства поиска. Архитектура нейронной сети состоит из операций: сверток, подвыборки, блоков (состоящих из нескольких слоев). Через OP_m обозначим множество всех операций, где m – ко-

личество операций. Перестановкой операций можно получить пространство поиска архитектур: $X = \{x_i | i = \overline{1, n}\}$, где через n обозначено количество архитектур. При этом, значение n , как правило, огромно. Например, при разработке архитектуры с 20 слоями пространство X состоит из m^{20} архитектур. Целью NAS является поиск оптимальной сети x^* , $x^* = f(X)$, в этом пространстве при использовании стратегии поиска f .

Для оптимизации поиска оптимальной сети стратегия поиска может включать предварительные знания, объем которых обозначается через G . Таким образом, задача формализуется так:

$$x^* = f(X, G).$$

Для эффективного включения G в стратегию поиска авторы работы [23] предложили использовать GPT, которая демонстрирует высокие результаты в решении задачи предсказания текста. Целью обучения модели GPT на крупномасштабном массиве ИНС является формирование у нее «общего понимания» нейросетевых архитектур и перенос его на конкретные задачи. Вся процедура поиска оптимальной архитектуры разделяется на 3 этапа: кодирование архитектуры ИНС, предварительное обучение и тонкая настройка модели GPT, поиск архитектуры ИНС с использованием обученной модели GPT.

Этап 1. Кодирование архитектуры ИНС. На этом этапе архитектура кодируется последовательностью символов, где каждый символ соответствует определенной операции (сверточный слой, полносвязный слой и т.д.) Эффективное кодирование архитектур нейронных сетей может помочь модели GPT усвоить фундаментальные законы архитектурной композиции. Поэтому необходимо разработать общую стратегию кодирования, подходящую для популярных архитектур ИНС. Основываясь на архитектурных закономерностях сверточных нейронных сетей (convolutional neural networks, CNN), авторы работы [23] разделяют слои на 4 категории: сверточные, подвыборки, полносвязные и другие. Код сверточного слоя должен включать его характеристики: размеры входа и выхода слоя, высоту и ширину ядра, величину шага (stride), ширину и способ обрамления (padding), величину расширения (dilation), количество групп каналов для обработки разными ядрами, возможность использования смещения (bias). Для кода подвыборки из ранее перечисленных характеристик следует исключить количество групп, но указать тип подвыборки (максимум или среднее). Полносвязный слой имеет только две характеристики: размеры входа и размеры выхода. К другим слоям относятся, например, слои активации, слои пакетной нормализации (batch normalization) и другие, для которых необходимо указать имя, размеры входа и выхода, а также значения используемых параметров. Таким образом, структура CNN кодируется текстовой строкой.

Этап 2. Предварительное обучение и тонкая настройка модели GPT. Предварительное обучение GPT выполняется на большом массиве данных, как правило, без учителя для достижения максимального правдоподобия. Тонкая настройка осуществляется с учителем на меньшем объеме данных. Но, учитывая, что на обоих этапах GPT-NAS обучается на архитектурах нейронных сетей, то на каждом этапе используется обучение с учителем. Целью обучения GPT-NAS является предсказание структурной информации о следующем слое по известной информации о предыдущих слоях. Таким образом, минимизируется следующая функция потерь:

$$\begin{cases} \mathfrak{F} = \sum_t^T \mathcal{L}(\hat{l}_t, l_t | l_{<t}, C), \\ C = f(\theta, S), \end{cases}$$

где через C обозначена архитектура ИНС, θ – набор значений ее параметров, S – соответствующая структура слоев. В формуле для целевой функции F через T обозначено количество слоев в C , L обозначает функцию потерь, \hat{l}_t и l_t – обозначают, соответственно, предсказанную и истинную структуру слоя с номером t , и $l_{<t} = (l_{t-k}, \dots, l_{t-1})$, k – размер окна данных.

Этап 3. Поиск архитектуры ИНС. Этот этап выполняется после предварительного обучения и тонкой настройки модели GPT. Этап состоит из двух шагов. Сначала архитектура отбирается, обучается и оценивается с использованием генетического алгоритма в качестве стратегии поиска. На втором шаге выбранные архитектуры реорганизуются с использованием GPT. На первом шаге каждой итерации инициализируются несколько архитектур. При этом инициализируемые архитектуры могут иметь разную длину. Затем оценивается эффективность каждой из этих архитектур. После этого выполняются эволюционные операции над получаемыми архитектурами, включая скрещивание, мутацию и отбор. Результатом первого шага является сеть с наилучшей эффективностью, полученная после выполнения заданного числа итераций. Второй шаг итерации. Соседние слои нейронной сети группируются в блоки. Блочная структура используется для ускорения процедуры селекции оптимальной архитектуры. Реконструкция сети на втором шаге итерации выполняется путем замены некоторого блока. Пусть, например, ИНС перед выполнением этого шага представлена последовательностью блоков в порядке их следования от входа сети к ее выходу: B_1, B_2, \dots, B_m . На этом шаге случайно выбирается некоторый слой оптимизируемой нейронной сети. Пусть этот слой содержится в блоке B_t . Из архитектуры удаляется этот блок B_t . Далее, с использованием модели GPT предсказывается слой l наиболее подходящий для последовательности слоев, образуемой блоками B_1, B_2, \dots, B_{t-1} . Предварительно авторами работы

[23] сформирован перечень из 12 стандартных блоков в привязке к стандартным архитектурам ИНС (таблица 1), которые они используют для замены блока B_t .

Таблица 1. Соответствие стандартных архитектур и блоков, содержащихся в них

Номер стандартной архитектуры ИНС	Стандартная архитектура ИНС	Наименование блока
1	EfficientNet	ConvNormActivation SqueezeExcitation
2	GoogleNet	Inception Avgpool
3	RegNet	ResBottleneckBlock Stem
4	ResNet	Bottleneck BasicBlock
5	ResNext	Bottleneck
6	ShuffleNet	InvertedResidual
7	Wide-ResNet	Bottleneck
8	other	Maxpool BatchNormal Relu Conv

С использованием предварительно обученной полносвязной ИНС классификации для слоя l выбирается наиболее подходящий блок B'_t , который вставляется на место B_t в последовательности B_1, B_2, \dots, B_m . Многократное повторение рассмотренной двухшаговой оптимизации позволяет достичь требуемого результата.

Разработанная авторами [23] система GPT-NAS сравнивалась с 20 современными конкурентами, среди которых были как алгоритмы, разработанные вручную, так и полученные с использованием NAS, на трех популярных наборах данных: CIFAR10, CIFAR100 и ImageNet-1K. Анализ экспериментальных результатов показал, что GPT-NAS достигает самых современных результатов, чем обосновывается положительное влияние GPT для разработки алгоритма NAS.

Вот пример архитектуры нейронной сети для бинарной классификации (лягушки или собаки) с использованием библиотеки Keras:

```
python
from keras.models import Sequential
from keras.layers import Dense, Flatten

# Создание модели
model = Sequential()

# Добавление слоев
model.add(Flatten(input_shape=(32, 32))) # Преобразование входных изображений в
одномерный вектор
model.add(Dense(64, activation='relu')) # Полносвязный слой с 64 нейронами и функцией
активации ReLU
model.add(Dense(1, activation='sigmoid')) # Выходной слой с 1 нейроном и функцией
активации sigmoid

# Компиляция модели
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

В данном примере используется последовательная модель Sequential, которая позволяет добавлять слои последовательно. Сначала добавляется слой Flatten, который преобразует входные изображения размером 32x32 в одномерный вектор. Затем добавляется полносвязный слой Dense с 64 нейронами и функцией активации ReLU. На выходе используется выходной слой Dense с 1 нейроном и функцией активации sigmoid, которая применяется для бинарной классификации.

Функция ошибки установлена как бинарная перекрестная энтропия (binary_crossentropy), а оптимизатор - Adam (adam). Метрика точности (accuracy) также указана для оценки производительности модели.

Рис. 1. Ответ чат-бота на запрос 1.

5. Эксперименты в Telegram с ChatGPT & Midjourney / AI bot

Запрос 1. Разработай в библиотеке Keras языка Python архитектуру нейронной сети бинарной классификации (лягушки или собаки). На вход сети подаются изображения размера 32 на 32. Сеть имеет 1 выход. Функция ошибки - бинарная перекрестная энтропия.

Ответ чат-бота представлен на рисунке 1.

Запрос 2 (уточнение). Добавь обучение на классах "лягушки" и "собаки" этой сети на базе CIFAR10.

Ответ чат-бота представлен на рисунках 2 и 3.

Результат проверки. Чат-ботом создана сеть бинарной классификации. В один из классов выделены лягушки, в другой объединены все остальные 9 классов базы CIFAR10, включая класс собак. Исправлять этот недочет не будем, а обучим модель и протестируем полученный

1. Загрузите данные CIFAR10 и разделите их на обучающую и тестовую выборки:

```
Python
from keras.datasets import cifar10
# Загрузка данных CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# Нормализация данных
x_train = x_train / 255.0
x_test = x_test / 255.0
# Преобразование меток в бинарные классы (0 - лягушка, 1 - собака)
y_train = (y_train == 6).astype(int)
y_test = (y_test == 6).astype(int)
```

2. Создайте модель и добавьте слои:

```
python
from keras.models import Sequential
from keras.layers import Dense, Flatten
# Создание модели
model = Sequential()
# Добавление слоев
model.add(Flatten(input_shape=(32, 32, 3))) # Преобразование входных изображений в
одномерный вектор
model.add(Dense(64, activation='relu')) # Полносвязный слой с 64 нейронами и функцией
активации ReLU
model.add(Dense(1, activation='sigmoid')) # Выходной слой с 1 нейроном и функцией
активации sigmoid
```

Рис. 2. Ответ чат-бота на запрос 2. Часть 1.

классификатор. Точность классификации на тестовом массиве после отработки скрипта составила 0.91.

Теперь проверим, как чат-бот отреагирует на более полный запрос, который включает запрос 1 и запрос 2.

Запрос 3. Напиши мне в библиотеке Keras языка Python программу разработки бинарного классификатора (лягушки или собаки) с использованием нейронной сети. На вход сети подаются изображения размера 32 на 32. Сеть имеет 1 выход. Функция ошибки - бинарная перекрестная энтропия. В программу включи возможность обучения и тестирования на классах "лягушки" и "собаки" этой сети на базе zifar10.

Ответ чат-бота представлен на рисунках 4 и 5.

Результат проверки. Чат-бот предложил существенно более сложную архитектуру нейронной сети, сохранив прежнее деление на 2 класса. Точ-

3. Компилируйте модель с функцией ошибки `binary_crossentropy`, оптимизатором `adam` и метрикой `accuracy`:

```
python
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

4. Обучите модель на обучающей выборке:

```
python
model.fit(x_train, y_train, batch_size=32, epochs=10, validation_data=(x_test, y_test))
```

5. Оцените производительность модели на тестовой выборке:

```
python
loss, accuracy = model.evaluate(x_test, y_test)
print('Test loss:', loss)
print('Test accuracy:', accuracy)
```

Вы можете настроить параметры обучения, такие как размер пакета (`batch_size`) и количество эпох (`epochs`), в зависимости от ваших потребностей.

Вы можете дальше настраивать эту архитектуру и обучать модель на своих данных.

Рис. 3. Ответ чат-бота на запрос 2. Часть 2.

```
python
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
# Загрузка данных CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# Преобразование меток в бинарные значения (0 для лягушек, 1 для собак)
y_train = (y_train == 6).astype(int)
y_test = (y_test == 6).astype(int)
# Нормализация данных
x_train = x_train / 255.0
x_test = x_test / 255.0
```

Рис. 4. Ответ чат-бота на запрос 3. Часть 1.

ность классификации новой сети на тестовой базе CIFAR10 оказалась существенно выше полученной в первом эксперименте и составила 0.95.

Попробуем добиться от сети корректного выделения изображений собак и лягушек.

Запрос 4. Напиши мне в библиотеке Keras языка Python программу разработки бинарного классификатора (лягушки или собаки) с исполь-

```

# Нормализация данных
x_train = x_train / 255.0
x_test = x_test / 255.0
# Создание модели нейронной сети
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Компиляция модели
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Обучение модели
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_test))
# Оценка производительности модели на тестовых данных
_, accuracy = model.evaluate(x_test, y_test)
print('Accuracy: %.2f' % (accuracy * 100))

```

Обратите внимание, что вам потребуется установить TensorFlow и Keras для запуска этого кода. Также обратите внимание, что это простой пример, и вам может потребоваться настроить модель и параметры обучения в соответствии с вашими потребностями и требованиями.

Рис. 5. Ответ чат-бота на запрос 3. Часть 2.

зованием нейронной сети. На вход сети подаются изображения размера 32 на 32 из базы CIFAR10. В обучающей и тестовой базах оставь только изображения лягушек и собак, остальные изображения отсей. Сеть имеет 1 выход. Функция ошибки - бинарная перекрестная энтропия. В программу включи возможность обучения и тестирования на классах "лягушки" и "собаки" этой сети на базе CIFAR10.

Ответ чат-бота представлен на рисунках 6 и 7.

Результат проверки. Чат-бот попытался выделить изображения лягушек и собак из обучающей и тестовой баз. При этом полученный код можно было использовать после некоторой доработки. Чат-ботом в этом эксперименте предложена упрощенная модель нейронной сети, ее точность на тестовом наборе составила всего 0.76.

Попытаемся повысить точность классификации. Для этого используем архитектуру, предложенную в ответе на запрос 3. В результате обу-

Ниже представлен пример программы на языке Python с использованием библиотеки Keras для разработки бинарного классификатора с использованием нейронной сети:

```
python
import numpy as np
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.utils import to_categorical
# Загрузка данных CIFAR-10
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
# Отфильтровать изображения, оставив только классы "лягушка" и "собака"
classes = [6, 5] # Лягушка - класс 6, Собака - класс 5
train_filter = np.isin(y_train, classes)
test_filter = np.isin(y_test, classes)
X_train = X_train[train_filter]
y_train = y_train[train_filter]
X_test = X_test[test_filter]
y_test = y_test[test_filter]
# Преобразование меток в формат one-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Рис. 6. Ответ чат-бота на запрос 4. Часть 1.

чения и тестирования получаем точность на тестовой базе 0.87. Можно ли увеличить этот показатель, используя чат-бот?

Запрос 5. У меня есть скрипт, позволяющий выполнять классификацию по двум классам (лягушки, собаки) на базе CIFAR10 с точностью 0.87. Привожу этот скрипт [здесь вставлен скрипт, содержащийся на рисунках 6 и 7]. Можно ли изменить архитектуру модели так, чтобы повысить точность классификации?

Ответ чат-бота. Да, можно изменить архитектуру модели для повышения точности классификации. Например, вы можете добавить дополнительные сверточные слои или полносвязные слои, увеличить количество нейронов в слоях, использовать различные функции активации, изменить параметры обучения, добавить регуляризацию или использовать другие методы оптимизации. Важно экспериментировать с различными архитектурами и параметрами модели, чтобы найти наилучшую комбинацию для данной задачи классификации.

```

# Преобразование изображений в формат, подходящий для обучения
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255
# Создание модели нейронной сети
model = Sequential()
model.add(Flatten(input_shape=(32, 32, 3)))
model.add(Dense(256, activation='relu'))
model.add(Dense(2, activation='softmax'))
# Компиляция модели
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Обучение модели
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32)
# Тестирование модели
_, accuracy = model.evaluate(X_test, y_test)
print(f"Точность классификатора: {accuracy}")

```

В данном примере используется нейронная сеть с одним скрытым слоем, состоящим из 256 нейронов. Функция активации для скрытого слоя - ReLU, а для выходного слоя - softmax. В качестве функции ошибки используется бинарная перекрестная энтропия. Для обучения модели используется оптимизатор Adam.

Программа загружает данные CIFAR10 и фильтрует изображения, оставляя только классы "лягушка" и "собака". Затем данные преобразуются в формат one-hot encoding и нормализуются. Модель нейронной сети создается и компилируется с заданными параметрами. После этого модель обучается на обучающем наборе данных и тестируется на тестовом наборе данных. В конце выводится точность классификатора на тестовом наборе данных.

Рис. 7. Ответ чат-бота на запрос 4. Часть 2.

Наш комментарий. Бот понял вопрос буквально и ограничился лишь рекомендациями, скрипт дорабатывать не стал.

Попробуем нагрузить его задачей детектирования.

Запрос 6. Предложи программу на Python с использованием архитектуры детектора YOLOv3 из модуля gluoncv, предобученного на базе COCO. Предусмотри обработку детектором случайно выбранного из проверочной части базы COCO изображения. Результат обработки (изображение и рамки) представь с использованием модуля matplotlib.

Ответ чат-бота представлен на рисунке 8.

Наша адаптация предложенного чат-ботом решения. Как и следовало ожидать, в этом случае пришлось выполнить существенную доработку кода. Его работоспособный вариант, использующий загруженные из COCO данные, после нашей доработки представлен на рисунках 9 и 10.

Конечно! Вот пример программы на Python, которая использует архитектуру детектора YOLOv3 из модуля gluoncv и предобученную на базе COCO модель. Она обрабатывает случайно выбранное изображение из проверочной части базы COCO и выводит результаты с использованием модуля matplotlib:

```
python
import matplotlib.pyplot as plt
from gluoncv import model_zoo, data, utils
# Загрузка предобученной модели YOLOv3
net = model_zoo.get_model('yolo3_darknet53_coco', pretrained=True)
# Загрузка данных из проверочной части базы COCO
test_dataset = data.COCODetection(splits=['instances_val2017'])
image, label = test_dataset[0]
# Обработка изображения с помощью модели YOLOv3
x, _ = data.transforms.presets.yolo.transform_test(image)
class_IDs, scores, bounding_boxes = net(x)
# Вывод изображения и рамок с помощью модуля matplotlib
ax = utils.viz.plot_bbox(image, bounding_boxes[0], scores[0], class_IDs[0],
class_names=net.classes)
plt.show()
```

Эта программа загружает предобученную модель YOLOv3 (yolo3_darknet53_coco) из модуля gluoncv и случайно выбирает одно изображение из проверочной части базы COCO. Затем она обрабатывает это изображение с помощью модели YOLOv3 и выводит результаты в виде изображения и рамок с использованием модуля matplotlib.

Обратите внимание, что для запуска этой программы вам понадобится установить модули gluoncv и matplotlib. Вы можете установить их с помощью pip:

```
pip install gluoncv matplotlib
```

Также убедитесь, что у вас есть доступ к базе COCO для загрузки данных.

Рис. 8. Ответ чат-бота на запрос 6.

Как видно, при доработке не решалась задача внесения минимальных исправлений для обеспечения работоспособности кода, а в код были внесены еще некоторые дополнения для анализа результатов работы сети.

Результат выполнения доработанного кода приведен на рисунке 11.

Выводы. Предложенные чат-ботом модули могут быть использованы как первичные грубые приближения для решения поставленной задачи, нуждающиеся в дальнейшей доработке экспертом. Ответы чат-бота не учитывают специфики версионности программных модулей, имеющегося у пользователя, могут ошибаться или неточно воспринимать запрос.

```

import matplotlib.pyplot as plt
from gluoncv import model_zoo, data, utils
import mxnet as mx
import time
import random
from pycocotools.coco import COCO
from pathlib import Path
from skimage import io
def test_model(model,img):
    # preprocessing
    mx_img = mx.nd.array(img)
    x,original_img = data.transforms.presets.rcnn.transform_test(mx_img)
    # time measurement
    start = time.time()
    box_ids, scores, bboxes = model(x)
    t = time.time() - start
    # image with model detections boxes, scores and ids
    ax = utils.viz.plot_bbox(original_img, bboxes[0], scores[0], box_ids[0],
class_names=model.classes)
    return ax,t

```

Рис. 9. Доработанный код. Часть 1.

```

yolo = model_zoo.get_model('yolo3_darknet53_coco', pretrained=True)
DATA_PATH = "D:/COCO/coco2017/val2017/"
annFile = Path("D:/COCO/coco2017/annotations/instances_val2017.json")
coco = COCO(annFile)
imgIds = coco.getImgIds()
img,anns = get_rand_img()
ax3,t3 = test_model(yolo,img)
ax3.set_title(f"YOLOv3 \n time taken {round(t3,4)}")
plt.show()

```

Рис. 10. Доработанный код. Часть 2.

6. Заключение

В этой работе выполнен обзор существующих подходов для решения задачи автоматизированного поиска архитектур нейронных сетей, удовлетворяющих запросу пользователя. Последние наиболее эффективные и перспективные решения связаны с применением больших языковых моделей, для обучения которых использовались значительные массивы данных. При этом применение больших языковых моделей для генерирования адекватных архитектур требует существенной алгоритмической доработки с применением, например, принципов генетического отбора и стратегии обучения с подкреплением.

Помимо обзора, выполнены некоторые эксперименты с ChatGPT & Midjourney доступного в Telegram для выяснения возможности использо-

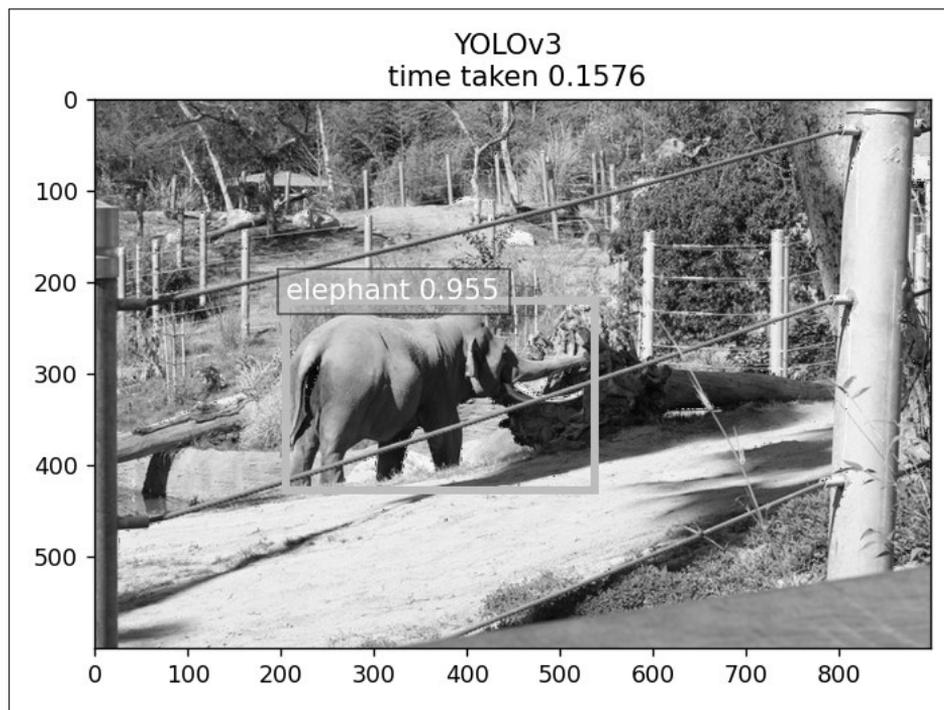


Рис. 11. Результат работы доработанного кода.

вания этого инструмента при разработке ИНС классификации и детектирования. Как показали результаты экспериментов, пока что от этого инструмента не следует ожидать окончательного точного решения, но его предложения можно использовать в качестве некоторой основы для последующей доработки.

Исследование выполнено при поддержке Министерства науки и высшего образования Российской Федерации (грант № 075-15-2020-801).

Список литературы

- [1] Elsken T., Metzen J. H., Hutter F., “Neural architecture search: A survey”, *The Journal of Machine Learning Research*, **20**:1 (2019), 1997–2017.
- [2] Zoph B., Le Q.V., “Neural architecture search with reinforcement learning”, *International Conference on Learning Representations*, 2017.
- [3] Bergstra J., Yamins D., Cox D. D., “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures”, *ICML*, 2013.

- [4] Mendoza H., Klein A., Feurer M., Springenberg J., Hutter F., “Towards Automatically-Tuned Neural Networks”, *International Conference on Machine Learning, AutoML Workshop*, 2016.
- [5] Kandasamy K., Neiswanger W., Schneider J. , Poczos B., Xing E. P., “Neural architecture search with Bayesian optimization and optimal transport”, *ArXiv: 1802.07191*, 2018.
- [6] Jaafra Y., Laurent J. L., Deruyver A., Naceur M. S., “Reinforcement learning for neural architecture search: A review”, *Image and Vision Computing*, **89** (2019), 57–66.
- [7] Liu Y., Sun Y., Xue B., Zhang M., Yen G. G., Tan K. C., “A survey on evolutionary neural architecture search”, *IEEE transactions on neural networks and learning systems*, 2021.
- [8] M. Wistuba, “Finding competitive network architectures within a day using UCT”, *ArXiv: 1712.07420*, 2017.
- [9] Letunovskiy A., Korviakov V., Polovnikov V., Kargapolitseva A., Mazurenko I., Xiong Y., “ISyNet: Convolutional Neural Networks design for AI accelerator”, *ArXiv: 2109.01932*, 2021.
- [10] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I., “Attention is all you need”, *ArXiv: 1706.03762v7*, 2018.
- [11] Radford A., Narasimhan K., Salimans T., Sutskever I., “Improving language understanding by generative pre-training, <https://gwern.net/doc/www/s3-us-west-2.amazonaws.com/d73fdc5ffa8627bce44dcda2fc012da638ffb158.pdf>”, 2018.
- [12] Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I., “Language models are unsupervised multitask learners”, *OpenAI blog*, **1:8** (2019).
- [13] Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A. et al., “Language models are few-shot learners”, *Advances in neural information processing systems*, **33** (2020), 1877–1901.
- [14] Chen M., Tworek J., Jun H., Yuan Q., Pinto H. P. d. O., Kaplan J., Edwards H., Burda Y., Joseph N., Brockman G. et al., “Evaluating large language models trained on code”, *ArXiv: 2107.03374*, 2021.

- [15] Todd G., Earle S., Nasir M. U., Green M. C., Togelius J., “Level generation through large language models”, *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 2023, 1–8.
- [16] Chen A., Dohan D. M., and So D. R., “Evoprompting: Language models for code-level neural architecture search”, *ArXiv: 2302.14838*, 2023.
- [17] Greydanus S., “Scaling down deep learning”, *ArXiv: 2011.14439*, 2020.
- [18] Veličković P., Badia A. P., Budden D., Pascanu R., Banino A., Dashevskiy M., Hadsell R., Blundell C., “The CLRS algorithmic reasoning benchmark”, *International Conference on Machine Learning. PMLR*, 2022, 84–102.
- [19] Caiyang Yu, Xianggen Liu, Wentao Feng, Chenwei Tang, Jiancheng Lv., “LLMatic: Neural Architecture Search via Large Language Models and Quality-Diversity Optimization”, *ArXiv: 2306.01102v1*, 2023.
- [20] Pugh J. K., Soros L. B., Stanley K. O., “Quality diversity: A new frontier for evolutionary computation”, *Frontiers in Robotics and AI*, 2016.
- [21] Mouret J.-B., Clune J., “Illuminating search spaces by mapping elites”, *ArXiv: 1504.04909*, 2015.
- [22] Stanley K. O., Miikkulainen R., “Evolving neural networks through augmenting topologies”, *Evolutionary computation*, **10**:2 (2002), 99–127.
- [23] Caiyang Yu, Xianggen Liu, Wentao Feng, Chenwei Tang, “GPT-NAS: evolutionary neural architecture search with generative pre-trained model”, *ArXiv: 2305.05351v2*, 2023.

Automation of artificial neural network architectures search
Chasovskikh A.A., Polovnikov V.S., Khusaenov A.A., Bokov G.V., Konovalov A.Yu, Drobyshev A.S., Biryukova V.A.

A review of works on methods for automating generation of artificial neural network architectures is presented. The most detailed material is presented on the use of large language models for this. For the experiments, we used ChatGPT & Midjourney, which is available in Telegram Messenger.

Keywords: artificial neural network, large language models, neural architecture search, generative pre-training transformer (GPT).

References

- [1] Elsken T., Metzen J. H., Hutter F., “Neural architecture search: A survey”, *The Journal of Machine Learning Research*, **20**:1 (2019), 1997–2017.
- [2] Zoph B., Le Q.V., “Neural architecture search with reinforcement learning”, *International Conference on Learning Representations*, 2017.
- [3] Bergstra J., Yamins D., Cox D. D., “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures”, *ICML*, 2013.
- [4] Mendoza H., Klein A., Feurer M., Springenberg J., Hutter F., “Towards Automatically-Tuned Neural Networks”, *International Conference on Machine Learning, AutoML Workshop*, 2016.
- [5] Kandasamy K., Neiswanger W., Schneider J. , Poczos B., Xing E. P., “Neural architecture search with Bayesian optimization and optimal transport”, *ArXiv: 1802.07191*, 2018.
- [6] Jaafra Y., Laurent J. L., Deruyver A., Naceur M. S., “Reinforcement learning for neural architecture search: A review”, *Image and Vision Computing*, **89** (2019), 57–66.
- [7] Liu Y., Sun Y., Xue B., Zhang M., Yen G. G., Tan K. C., “A survey on evolutionary neural architecture search”, *IEEE transactions on neural networks and learning systems*, 2021.
- [8] M. Wistuba, “Finding competitive network architectures within a day using UCT”, *ArXiv: 1712.07420*, 2017.
- [9] Letunovskiy A., Korviakov V., Polovnikov V., Kargapol'tseva A., Mazurenko I., Xiong Y., “ISyNet: Convolutional Neural Networks design for AI accelerator”, *ArXiv: 2109.01932*, 2021.
- [10] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I., “Attention is all you need”, *ArXiv: 1706.03762v7*, 2018.
- [11] Radford A., Narasimhan K., Salimans T., Sutskever I., “Improving language understanding by generative pre-training”, <https://gwern.net/doc/www/s3-us-west-2.amazonaws.com/d73fdc5ffa8627bce44dcda2fc012da638ffb158.pdf>, 2018.

- [12] Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I., “Language models are unsupervised multitask learners”, *OpenAI blog*, **1**:8 (2019).
- [13] Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A. et al., “Language models are few-shot learners”, *Advances in neural information processing systems*, **33** (2020), 1877–1901.
- [14] Chen M., Tworek J., Jun H., Yuan Q., Pinto H. P. d. O., Kaplan J., Edwards H., Burda Y., Joseph N., Brockman G. et al., “Evaluating large language models trained on code”, *ArXiv: 2107.03374*, 2021.
- [15] Todd G., Earle S., Nasir M. U., Green M. C., Togelius J., “Level generation through large language models”, *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 2023, 1–8.
- [16] Chen A., Dohan D. M., and So D. R., “Evoprompting: Language models for code-level neural architecture search”, *ArXiv: 2302.14838*, 2023.
- [17] Greydanus S., “Scaling down deep learning”, *ArXiv: 2011.14439*, 2020.
- [18] Veličković P., Badia A. P., Budden D., Pascanu R., Banino A., Dashevskiy M., Hadsell R., Blundell C., “The CLRS algorithmic reasoning benchmark”, *International Conference on Machine Learning. PMLR*, 2022, 84–102.
- [19] Caiyang Yu, Xianggen Liu, Wentao Feng, Chenwei Tang, Jiancheng Lv., “LLMatic: Neural Architecture Search via Large Language Models and Quality-Diversity Optimization”, *ArXiv: 2306.01102v1*, 2023.
- [20] Pugh J. K., Soros L. B., Stanley K. O., “Quality diversity: A new frontier for evolutionary computation”, *Frontiers in Robotics and AI*, 2016.
- [21] Mouret J.-B., Clune J., “Illuminating search spaces by mapping elites”, *ArXiv: 1504.04909*, 2015.
- [22] Stanley K. O., Miikkulainen R., “Evolving neural networks through augmenting topologies”, *Evolutionary computation*, **10**:2 (2002), 99–127.
- [23] Caiyang Yu, Xianggen Liu, Wentao Feng, Chenwei Tang, “GPT-NAS: evolutionary neural architecture search with generative pre-trained model”, *ArXiv: 2305.05351v2*, 2023.

Часть 2.
Специальные вопросы теории
интеллектуальных систем

Об автоматных неисправностях при алфавитном кодировании

П. С. Дергач¹ Д. Б. Бахрамова¹

В данной работе исследован вопрос о возможных неисправностях регулярных языков, сохраняющих свойство неоднозначности алфавитного кодирования. Показано, что любая диаграмма Мура, задающая неоднозначно кодируемый язык, всегда может сохранить свойство неоднозначности при разовой неисправности выходного значения на стрелке.

Ключевые слова: алфавитное кодирование, диаграмма Мура, автоматные неисправности.

1. Введение

В настоящее время автоматы, использующие алфавитное кодирование, широко применяются в различных областях, таких как телекоммуникации, компьютерные системы, промышленность и другие. Однако при использовании таких автоматов могут возникать неисправности, которые могут привести к сбоям в работе системы.

Одним из ключевых элементов обработки информации является кодирование, которое позволяет преобразовывать информацию из одной формы в другую, сохраняя при этом целостность и точность. Однако, при использовании различных методов кодирования могут возникать проблемы, такие как потеря однозначности или возникновение неоднозначности при восстановлении схем кодирования.

Целью данной статьи является описание класса диаграмм Мура, которые могут сохранить свойство неоднозначности декодирования при разовой неисправности.

Для желающих подробнее изучить проблематику алфавитного кодирования рекомендуются к прочтению работы [6-8]. С проблематикой тестирования и диагностики неисправностей логических устройств можно ознакомиться в работах [9-11].

¹ *Дергач Пётр Сергеевич* — к.ф.-м.н., м.н.с. каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: dergachpes@gmail.com.

Dergach Peter Sergeevich — Ph.D., junior researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

² *Бахрамова Дильяра Бахрамовна* — выпускник Филиала МГУ имени М. В. Ломоносова в городе Ташкенте, e-mail: bkhr.lia.1807@gmail.com.

Bakhranova Dilyara Bakhramovna — Graduate of the M. V. Lomonosov Moscow State University Branch in Tashkent.

2. Постановка задачи

Рассматриваем Диаграмму Мура с входным и выходным алфавитами $0, 1$ и функцию алфавитного кодирования с теми же входным и выходным алфавитами. Пусть эта Диаграмма Мура распознает по выходному символу 1 регулярный язык P и f неоднозначно на P . Требуется определить, всегда ли найдется единичная неисправность выходного значения на стрелке диаграммы, для которой новый распознаваемый по выходному символу 1 регулярный язык по-прежнему неоднозначен на f .

3. Основные определения и формулировки

Определения этого раздела взяты по большей части из источников [1-5].

Пусть C - некоторое конечное множество, называемое *алфавитом*. Элементы этого множества называем *буквами*. Если $\gamma = c(1)...c(n)$ - конечная последовательность букв алфавита C , то говорим, что γ есть *слово* в алфавите C . Число n называем *длиной слова* γ и обозначаем $|\gamma|$. *Абстрактным конечным автоматом* называется набор

$$V = (A, Q, B, \phi, \psi),$$

где A, Q, B - конечные множества, ϕ - функция, определенная на множестве $Q \times A$ и принимающая значения из Q , ψ - функция, определенная на множестве $Q \times A$ и принимающая значения из B . Множества A, Q, B называются соответственно *входным алфавитом*, *алфавитом состояний* и *выходным алфавитом* автомата V . Функция ϕ - называется *функцией переходов*, а функция ψ - *функцией выходов автомата* V .

Пусть $V = (A, Q, B, \phi, \psi)$ - конечный абстрактный автомат. Для каждого состояния q автомата V можно рассмотреть набор (A, Q, B, ϕ, ψ, q) , определяющий автомат V с выделенным начальным состоянием q . Такие наборы (A, Q, B, ϕ, ψ, q) называются *инициальными конечными абстрактными автоматами*.

Класс инициальных конечных абстрактных автоматов с входным и выходным алфавитами $\{0, 1\}$ обозначаем через K_2 .

Функции переходов и выходов алфавита

$$V = (A, Q, B, \phi, \psi, q)$$

доопределим на множестве $Q \times A^*$ (сохраним за ними те же обозначения). Именно, полагаем по определению

$$\phi(q, \alpha a) = \phi(\phi(q, \alpha), a),$$

где $q \in Q, \alpha \in A^*, a \in A$. Аналогично,

$$\psi(q, \alpha a) = \psi(\phi(q, \alpha), a).$$

Автомат

$$V = (\{0, 1\}, Q, \{0, 1\}, \phi, \psi, q)$$

допускает слово α , если при подаче на вход α последний символ слова $\phi(q, \alpha)$ соответствующего выходного слова равен 1. Множество слов, допускаемых автоматом, обозначаем через $1(V)$ и называем его *распознаваемым автоматом*.

Диаграммой Мура автомата

$$V = (\{0, 1\}, Q, \{0, 1\}, \phi, \psi, q)$$

называется ориентированный граф G , вершинами которого являются состояния $q \in Q$ и для каждого равенства вида $\phi(a, q) = q^*$ граф имеет ребро, идущее из q в q^* , на котором стоит метка a, b , где $b = \psi(q, a)$. Начальное состояние q помечается на диаграмме звездочкой.

Для построения диаграммы Мура для каждого состояния q_i алфавита $Q = \{q_1, \dots, q_n\}$ на плоскости размещается свой круг, внутри каждого круга пишется символ q_i . Далее рассматриваются всевозможные пары (q_i, a_j) , где $q_i \in Q, a_j \in A, A = \{0, 1\}$. Для каждой такой пары от круга, в котором записан символ q_i , проводится стрелка к кругу, в котором записан символ $\phi(q_i, a_j)$. Этой стрелке приписывается пара $(a_j, \psi(q_i, a_j))$.

Пусть $A = \{a_1, \dots, a_r\}$ – произвольный конечный непустой алфавит. Пусть P_1, P_2 – непустые множества слов в алфавите A . Определим следующие операции над P_1 и P_2 .

- 1) *Объединение* множеств P_1 и P_2 (обозначим $P_1 \cup P_2$) есть множество всех слов вида α , где $\alpha_1 \in P_1$ или $\alpha_2 \in P_2$.
- 2) *Итерация* множества P_1 (обозначаем $(P_1)^*$) есть множество всех слов вида $\alpha_1, \dots, \alpha_k$, где $\alpha_1 \in P_1, \dots, \alpha_k \in P_1, k \geq 1$.
- 3) *Конкатенация* множеств P_1 и P_2 (обозначаем $P_1 \cdot P_2$) есть множество всех слов вида $\alpha_1 \alpha_2$, где $\alpha_1 \in P_1, \alpha_2 \in P_2$.

Введем понятие регулярного множества в алфавите A . Множество P называется *регулярным в алфавите A* , если его можно получить из пустого множества и одноэлементных однобуквенных множеств $a, a \in A$, применением конечного числа конкатенаций, объединений и итераций. Более подробно, определение регулярных множеств таково:

- 1) \emptyset – регулярное множество в алфавите A ;
- 2) a , где a – произвольная буква алфавита A , – регулярное множество в алфавите A ;
- 3) Если P_1, P_2 – регулярные множества в алфавите A , то и множества $P_1 \cup P_2, P_1 \cdot P_2, (P_1)^*$ – регулярные множества в алфавите A ;
- 4) Регулярность произвольного множества в алфавите A устанавливается в соответствии с пунктами (1)-(3) за конечное число шагов.

Отображение $f : \{0, 1\} \rightarrow \{0, 1\}^*$ следующего вида

$$f(0) = \alpha,$$

$$f(1) = \beta$$

называем *схемой кодирования из алфавита $\{0, 1\}$ в алфавит $\{0, 1\}$* . Множество всех схем кодирования из алфавита $\{0, 1\}$ в алфавит $\{0, 1\}$ обозначаем через F_2 .

Пусть β – непустое слово в алфавите A . Если существует слово α в алфавите A , для которого при некотором $k \in \mathbb{N}$ имеем $\beta = \alpha^k$, то называем слово α *измельчением слова β* .

Пусть $f \in F_2$ и слова $f(0), f(1)$ соизмеримы. Называем класс таких схем соизмеримым и обозначаем его через \hat{F}_2 .

Доопределим отображение $f \in F_2$ до отображения $\tilde{f} : \{0, 1\}^* \rightarrow \{0, 1\}^*$ следующим образом:

$$\tilde{f}(a_{i_1}a_{i_2}\dots a_{i_n}) = f(a_{i_1})f(a_{i_2})\dots f(a_{i_n}).$$

Отображение \tilde{f} называется *алфавитным кодированием из алфавита $\{0, 1\}$ в алфавит $\{0, 1\}$* .

Пусть $f \in F_2$ и $P \subseteq \{0, 1\}^*$ – регулярный язык. Кодирование \tilde{f} называется *взаимно однозначным* на P , если разные слова из P переходят в разные слова.

Пусть G – диаграмма Мура. *Единичной неисправностью выхода* называем замену выходного значения $\phi(q, a)$ на одном из ребер диаграммы (0 заменяется на 1, 1 заменяется на 0).

Пусть $f \in F_2$ и G – диаграмма Мура, для автомата $V \in K_2$, причем f неоднозначно на $1(V)$. Пусть ρ – ребро диаграммы G . Говорим, что единичная неисправность выхода на ρ *сохраняет свойство неоднозначности на f* , если f неоднозначно на языке, задаваемом диаграммой после неисправности.

Теорема 1. Пусть G – диаграмма Мура автомата $V \in K_2$ и пусть $f \in F_2$ – алфавитное кодирование, неоднозначное на $1(V)$. Тогда в G найдется такое ребро, замена выходного значения которого сохраняет свойство неоднозначности на f .

4. Доказательства вспомогательных утверждений

Лемма 1. Пусть A – конечный алфавит, $\alpha, \beta \in A^* \setminus \{\Lambda\}$ и $\alpha^k = \beta^m$ для некоторых $k, m \in \mathbb{N}$. Тогда существует $\nu \in A^* \setminus \{\Lambda\}$ такое, что

$$|\nu| = (|\alpha|, |\beta|), \alpha = \nu^{|\alpha|/|\nu|}, \beta = \nu^{|\beta|/|\nu|}.$$

Доказательство. Доказательство леммы приведено во второй главе [1]. □

Лемма 2. Пусть $f \in F_2$ и $f(0) = \alpha$, $f(1) = \beta$. Тогда f неоднозначно на $\{0, 1\}^*$ тогда и только тогда, когда α и β соизмеримы.

Доказательство. Доказательство леммы приведено в [2] (Лемма 10). □

Лемма 3. Если язык $P \subseteq \{0, 1\}^*$ неоднозначен на $f \in F_2$, то при единичной несправности замены выходного значения $0 \rightarrow 1$ сохраняется неоднозначность.

Доказательство. В неоднозначном регулярном языке, который мы рассматриваем изначально, есть пара различных слов, имеющих одинаковый код. При расширении языка это свойство сохраняется. □

Лемма 4. Если язык $P \subseteq \{0, 1\}^*$ однозначен на $f \in F_2$, то при единичной несправности замены выходного значения $1 \rightarrow 0$ сохраняется однозначность.

Доказательство. В однозначном регулярном языке, который мы рассматриваем изначально, нет пары различных слов, имеющих одинаковый код. При сужении языка это свойство сохраняется. □

Лемма 5. Пусть $f \in \hat{F}_2$, $V \in K_2$ и G – его диаграмма Мура, причем все выходные значения на стрелках в ней равны 1. Пусть в G есть достижимое из начальной вершины q_0 состояние q , через которое проходит пара циклов с входными значениями слов α и β , причем эта пара слов несоизмерима. Тогда \tilde{f} не взаимно однозначно на $1(V)$.

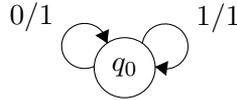
Доказательство. Пусть γ – слово в алфавите $\{0, 1\}$, при подаче которого на q_0 автомат переходит в состояние q . Тогда множества $\gamma\alpha^*$ и $\gamma\beta^*$ лежат в $1(V)$, так как все выходные значения диаграммы равны 1. Так как $f(0)$ и $f(1)$ соизмеримы, то

$$\begin{aligned} f(0) &= \nu^r, \\ f(1) &= \nu^s \end{aligned}$$

для некоторых $\nu \in \{0, 1\}^*$, $r, s \in \mathbb{N}$. Тогда $\tilde{f}(\alpha) = \nu^{r1}$, $\tilde{f}(\beta) = \nu^{s1}$ для некоторых $r_1, s_1 \in \mathbb{N}$. В этом случае слова $\gamma\alpha^{s_1}$ и $\gamma\beta^{r_1}$ имеют одинаковый код. Но они не равны, так как иначе они были бы соизмеримы (по Лемме 1). \square

Лемма 6. Пусть $f \in \hat{F}_2$, $V \in K_2$ и G – его диаграмма Мура, имеющая не больше двух ребер и причем все выходные значения диаграммы равны 1. Тогда в G существует единичная неисправность выхода, сохраняющая свойство неоднозначности на f .

Доказательство. Заметим, что единственная диаграмма Мура, имеющая не более двух ребер, выглядит так:



Поэтому $1(V) = \{0, 1\}^*$. Очевидно, что f неоднозначна на $1(V)$, так как для $f \in \hat{F}_2$ имеем

$$\begin{aligned} f(0) &= \nu^r, \\ f(1) &= \nu^s \end{aligned}$$

для некоторых $\nu \in \{0, 1\}^*$, $r, s \in \mathbb{N}$. И $\tilde{f}(01) = \tilde{f}(10)$.

Покажем теперь верность основного утверждения леммы. Применим единичную неисправность к выходу ребра с входным значением 0. Получим диаграмму, задающую множество $\{0, 1\}^*1$. Так как $\tilde{f}(011) = \tilde{f}(101)$, то наша неисправность сохраняет свойство неоднозначности на f . Утверждение леммы доказано. \square

5. Доказательства основных утверждений

Теорема 1. Пусть G – диаграмма Мура автомата $V \in K_2$ и пусть $f \in F_2$ – алфавитное кодирование, неоднозначное на $1(V)$. Тогда в G найдется такое ребро, замена выходного значения которого сохраняет свойство неоднозначности на f .

Доказательство. Если в G есть хотя бы одно ребро с выходным значением 0, то при единичной неисправности на таком ребре язык расширится и по лемме 3 получаем сохранение свойства однозначности на f .

Далее будем считать, что все ребра в G имеют выходные значения 1. Будем подавать последовательность из 0 на начальное состояние диаграммы G . В какой-то момент мы вернемся к вершине (назовем ее q_1), в которой были ранее, то есть образуется цикл. Теперь на q_1 подаем последовательность из 1. В какой-то момент мы встретим вершину (назовем ее q_2), которая была ранее на 0 или вершину (назовем ее q_3), которая была на 1.

Рассмотрим случай, когда вернулись на вершину q_2 и эта вершина была в последовательности по 0 до вершины q_1 (показано на Рис. 1).

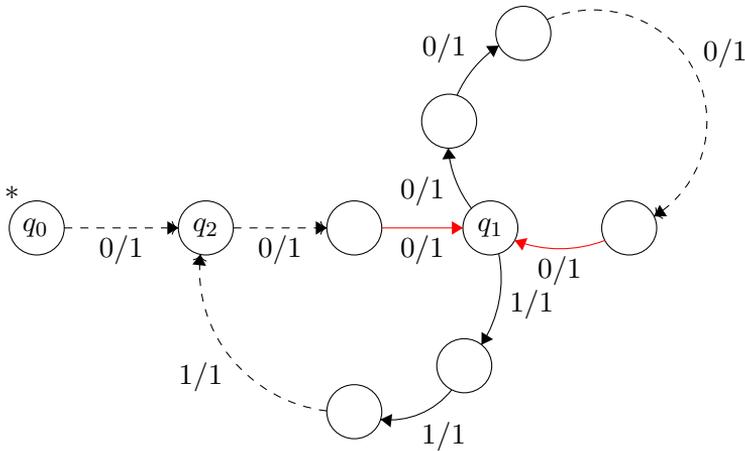


Рис.1.

Тогда через вершину q_1 проходит два цикла, причем в одном из циклов подаются только 0, а в другом – и 0, и 1. Поэтому в силу леммы 5 \tilde{f} неоднозначно на $1(V)$. Чтобы это свойство сохранилось, нам достаточно применить единичную неисправность к ребру, которое отлично от последних ребер циклов. Это всегда можно сделать, если в диаграмме G есть хотя бы три ребра. Если же в G нет трех ребер, то утверждение следует из леммы 6.

Рассмотрим случай, когда мы вернулись в цикл по 0 (показано на Рис. 2).

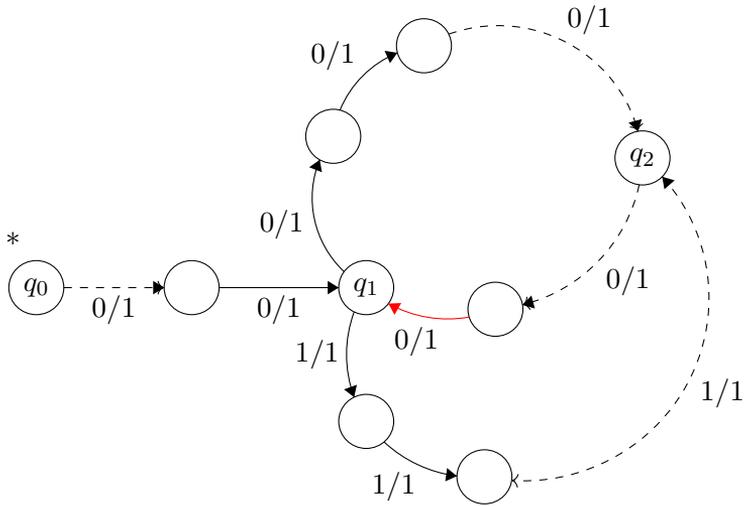


Рис.2.

Тогда через вершину q_1 проходит два цикла, причем в одном из циклов подаются только 0, а в другом – и 0, и 1. или только 1 (когда $q_1 = q_2$). Далее рассуждение полностью повторяет доказательство предыдущего случая.

Рассмотрим случай, когда вернулись на вершину по 1 (показано на Рис.3.)

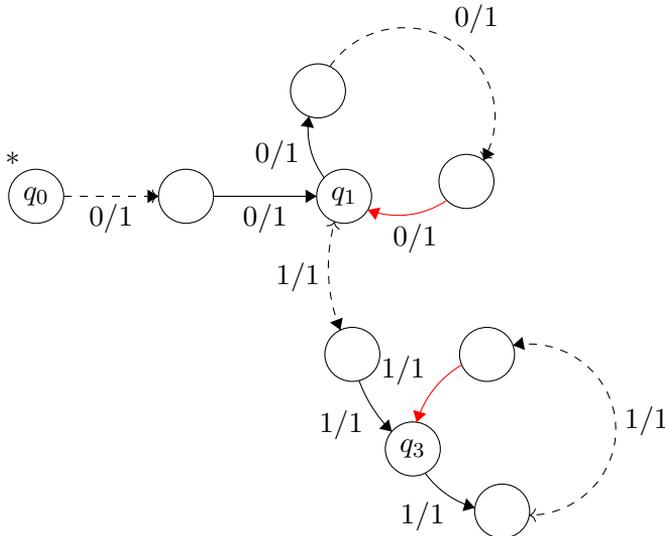


Рис.3.

Тогда подадим последовательность из 0 на вершину q_3 . Здесь опять возможны три случая. Либо мы вернемся к вершине, которая была до последнего цикла из 1, либо мы вернемся к вершине, которая была на

этом цикле, либо мы образуем новый цикл. В первых двух случаях рассуждения полностью аналогичны рассуждениям, приводимым выше (с образованием двух циклов, один из которых состоит или только из 0, или только из 1, а другой – либо и из 0, и из 1, либо только из противоположного бита к первому циклу). А в третьем случае, образуется новый цикл по 0. В какой-то момент новые циклы образовываться перестанут и рассуждение завершится. Утверждение доказано. \square

6. Заключение

В данной статье было исследовано важное свойство автоматов, связанное с неоднозначностью алфавитного кодирования на диаграммах Мура. Было установлено, что для произвольной диаграммы Мура G автомата $V \in K_2$ и для произвольного алфавитного кодирования $f \in F_2$, неоднозначного для $1(V)$, всегда найдется единичная неисправность выходного значения на ребре, сохраняющая свойство неоднозначности на f .

Дальнейшим направлением исследования могло бы послужить нахождение классов диаграмм Мура, сохраняющих свойство однозначности хотя бы при одной единичной неисправности. Также интерес представляет исследование для случая, когда рассматривается свойство однозначности/неоднозначности при произвольной единичной неисправности.

Список литературы

- [1] Дергач П.С., “Алфавитное кодирование регулярных языков с полиномиальной функцией роста.”, *Кандидатская диссертация.*, 2016., стр. 1–213..
- [2] Дергач П.С., “О проблеме вложения допустимых классов.”, *Интеллектуальные системы, изд. МГУ, М.*, том 19, № 2. (2015.), стр. 143–174..
- [3] Дергач П.С., “Об однозначности алфавитного декодирования.”, *Дискретная математика -М.: Наука.*, том 24, № 4, (2012.), стр. 80–90..
- [4] В. Б. Кудрявцев, С. В. Алешин, А. С. Подколзин., “Введение в теорию автоматов.”, *М.: Наука*, 1985..
- [5] Яблонский С.В., “Введение в дискретную математику.”, *М.: Наука.*, 1986..
- [6] Dergach P.S., “On uniqueness of alphabetical decoding of ϵ -regular languages.”, *Discrete Mathematics and Applications, издательство V S P (Netherlands).*, том 24, № 3, (2014.), стр. 139–152..

- [7] Марков А. А., “Введение в теорию кодирования.”, *М: Наука*, 1982..
- [8] Жильцова П. Л., “Современные проблемы теории кодирования.”, *Учебное пособие, Нижний Новгород.*, 2007..
- [9] Редькин П. Н., “Надёжность и диагностика схем.”, *М.: Изд-во МГУ*, 1992., стр. 192..
- [10] Попков А. К., “Нижние оценки длин единичных тестов для схем из функциональных элементов.”, *Дискретная математика.*, **том. 29. Вып. 2.** (2017.), стр. 53–69..
- [11] Романов С. Д., “Метод синтеза легкотестируемых схем, допускающих единичные проверяющие тесты константной длины.”, *Discrete Mathematics.*, **том. 26. Вып. 2.** (2014.), стр. 100–130..

About automaton malfunctions in alphabetical coding
Dergach P.S., Bahramova D.B.

The question of possible faults in regular languages that preserve the property of ambiguity in alphabet coding is investigated in this work. It is shown that any Moore diagram that defines a language with ambiguous encoding can always maintain the property of ambiguity in the event of a one-time fault in the output value on an arrow.

Keywords: alphabet encoding, Moore diagram, automaton faults.

References

- [1] Dergach P.S., “Alphabetic encoding of regular languages with polynomial growth function”, *PhD thesis*, 2016, 1–213.
- [2] Dergach P.S., “About the problem of embedding permissible classes.”, *Intelligent Systems, published by MSU, Moscow.*, **Volume 19 No. 2.** (2019), 143–174.
- [3] Dergach P.S., “On the unambiguity of alphabet decoding.”, *Discrete Mathematics - Moscow: Nauka*, **Volume 24 No. 4.** (2012), 80–90.
- [4] Kudryavtsev V.B., Aleshin S.V., Podkolzin A.S., “Introduction to automata theory”, 1985.
- [5] Yablonskiy S.V., “Introduction to discrete math”, 1986.
- [6] Dergach P.S., “On uniqueness of alphabetical decoding of ϵ -regular languages”, *Discrete Mathematics and Applications, publisher V S P (Netherlands)*, **Volume 24, No. 3.** (2014), 139–152.

- [7] Markov A. A., “Introduction to Coding Theory”, *Moscow: Nauka.*, 1982.
- [8] Zhiltsova P. L., “Introduction to Coding Theory”, *Educational Manual, Nizhny Novgorod.*, 2007.
- [9] Redkin P. N., “Reliability and Fault Diagnosis of Circuits.”, *Moscow: Moscow State University Publishing.*, 1992, 192.
- [10] Popkov A. K., “Lower Bounds on the Lengths of Unit Tests for Circuits Composed of Functional Elements.”, *Discrete Mathematics.*, **Volume 29. Issue 2.** (2017), 53–69.
- [11] Romanov S. D., “Method of Synthesis of Easily Testable Circuits Allowing Unit-Length Checking Tests.”, *Discrete Mathematics.*, **Volume 26. Issue 2.** (2014.), 100–130.

Сложность задачи о существовании сюръективного гомоморфизма на рефлексивные циклы

Н. П. Корчагин¹

В работе исследуется сложность массовой задачи, где на вход подается произвольный граф, и нужно определить, существует ли сюръективный гомоморфизм из этого графа в фиксированный граф \mathcal{G} . Мы доказываем, что если \mathcal{G} — рефлексивный цикл длины 7 и более, то задача является NP-трудной.

Доказательство основано на новом подходе, который позволяет сводить решение задачи удовлетворения ограничениям к проверке выполнимости системы тождеств, а она сводится к решению сюръективной задачи удовлетворения ограничениям.

Ключевые слова: сюръективные гомоморфизмы, сложность вычислений, удовлетворение ограничениям, полиморфизмы.

1. Введение

В данной статье мы рассмотрим задачу о существовании сюръективного гомоморфизма на заданный граф (*Surj-Hom*) и ее связь с задачей об удовлетворении ограничениям (CSP), сюръективном удовлетворении ограничениям (SCSP).

Рассмотрим граф $\mathcal{G} = (V, E)$ с множеством вершин V и множеством ребер $E \subseteq V \times V$ и граф $\mathcal{H} = (V', E')$ с вершинами V' и ребрами $E' \subseteq V' \times V'$. *Петлей* в графе называется ребро, которое соединяет вершину с ней самой. Граф называется *рефлексивным*, если каждая его вершина содержит петлю. *Гомоморфизм* графа \mathcal{H} на граф \mathcal{G} — это отображение $f : V' \rightarrow V$ такое, что $\forall (a_i, a_j) \in E'$ верно, что $(f(a_i), f(a_j)) \in E$. Для фиксированного графа \mathcal{H} задача о существовании сюръективного гомоморфизма *Surj-Hom*(\mathcal{H}) — это массовая задача, в которой по данному графу \mathcal{G} требуется проверить, существует ли сюръективный гомоморфизм \mathcal{G} на \mathcal{H} . C_n^{ref} — это рефлексивный цикл длины n .

Основным результатом статьи является следующая теорема:

Теорема 1. *Surj-Hom*(C_n^{ref}) является NP-трудной при $n \geq 7$.

Ниже мы формально определим понятия, которые будут использоваться в доказательстве этой теоремы.

¹Корчагин Никита Павлович — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: kolkor92@gmail.com.

Korchagin Nikita Pavlovich — graduate student, Lomonosov State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

1.1. Основные понятия

Задача *Constraint Satisfaction Problem* (CSP) - известная массовая задача, в которой необходимо определить, существует ли подстановка, удовлетворяющая определенному набору ограничений. Формально она определяется следующим образом: пусть D – конечное множество (*область значений*), Γ – конечное множество отношений на D . Тогда задача удовлетворения ограничениям, также обозначаемая как $\text{CSP}(\Gamma)$ – это задача, в которой по формуле \mathcal{I}

$$R_1(z_{1,1}, \dots, z_{1,n_1}) \wedge \dots \wedge R_s(z_{s,1}, \dots, z_{s,n_s}),$$

где $R_1, \dots, R_s \in \Gamma$ – отношения из Γ и $z_{i,j} \in \{x_1, \dots, x_n\}$ – переменные над D , необходимо определить, выполнима ли она. Данная формула вместе с набором переменных $\{x_1, \dots, x_n\}$ называется *экземпляром* \mathcal{I} задачи $\text{CSP}(\Gamma)$, а подстановка, удовлетворяющая всем ограничениям – *решением* экземпляра.

k -местная функция f называется *полиморфизмом* m -местного отношения R , если для любых $(a_1^1, \dots, a_1^m), \dots, (a_k^1, \dots, a_k^m)$ из R , верно, что $(f(a_1^1, \dots, a_k^1), \dots, f(a_1^m, \dots, a_k^m))$ тоже из R (говорят также, что f *сохраняет* R). f – полиморфизм множества отношений Γ , если f – полиморфизм каждого отношения из Γ . Множество полиморфизмов Γ обозначается как $\text{Pol}(\Gamma)$. Множество сюръективных полиморфизмов Γ обозначается как $\text{SPol}(\Gamma)$.

Функция f от n переменных называется *существенно-унарной*, если она существенно зависит не более, чем от одной переменной. Иными словами, f – существенно унарна тогда и только тогда, когда существуют $i \in \{1, \dots, n\}$, g – функция от одной переменной такие, что $f(x_1, \dots, x_i, \dots, x_n) = g(x_i)$.

В задаче *Surjective Constraint Satisfaction Problem* (SCSP) помимо выполнимости формулы также требуется сюръективность решения. Эта задача определяется так. Пусть Γ – конечное множество отношений на области значений D . В задаче SCSP(Γ) по данной формуле \mathcal{I}

$$\mathcal{I} = R_1(z_{1,1}, \dots, z_{1,n_1}) \wedge \dots \wedge R_s(z_{s,1}, \dots, z_{s,n_s}),$$

где $R_1, \dots, R_s \in \Gamma$ и $z_{i,j} \in \{x_1, \dots, x_n\}$ необходимо определить, существует ли решение такое, что $\{x_1, \dots, x_n\} = D$. Покажем, что *Surj-Hom*(\mathcal{H}) эквивалентен SCSP(E), где E – бинарное отношение смежности на вершинах графа \mathcal{H} . Для графа $\mathcal{H} = (V, E)$ с $V = \{v_1, \dots, v_n\}$ рассмотрим отношение смежности на множестве его вершин $E \subseteq V \times V$. Тогда существование сюръективного гомоморфизма произвольного графа $\mathcal{G} = (V', E')$ с $V' = \{x_1, \dots, x_m\}$ и $E' = \{(x_{1_1}, x_{1_2}), (x_{2_1}, x_{2_2}), \dots, (x_{l_1}, x_{l_2})\}$ на \mathcal{H} эквивалентно существованию сюръективного решения формулы

$$E(x_{1_1}, x_{1_2}) \wedge E(x_{2_1}, x_{2_2}) \wedge \dots \wedge E(x_{l_1}, x_{l_2}).$$

Аналогично, если E – некоторое бинарное отношение на множестве $V = \{v_1, \dots, v_n\}$, то по произвольной формуле

$$E(z_{1_1}, z_{1_2}) \wedge \dots \wedge E(z_{s_1}, z_{s_2}),$$

где $z_{i_j} \in \{x_1, \dots, x_m\}$ можно построить граф \mathcal{H} с t вершинами x_1, \dots, x_m и s ребрами (z_{i_1}, z_{i_2}) . Тогда каждое сюръективное решение формулы соответствует некоторому сюръективному гомоморфизму из \mathcal{H} на граф с вершинами $\{v_1, \dots, v_n\}$ и отношением смежности E .

1.2. Предыдущие результаты

Многие комбинаторные задачи можно представить в виде CSP для некоторых Γ [11]. В [19] было предложено и, позже, независимо доказано в [4] и [5], что для произвольного Γ эта задача либо решается за полиномиальное время, либо является NP-полной. Более того, в [4] и [5] была независимо описана полная классификация сложности CSP для произвольных Γ , которая сформулирована с помощью полиморфизмов Γ .

После получения классификации сложности CSP, интерес исследователей сместился в сторону различных вариаций этой задачи, таких как SCSP. Известно, что если Γ – набор отношений на множестве D , то SCSP(Γ) можно свести к CSP($\Gamma \cup \{x = d \mid d \in D\}$) [1]. Так как сложность CSP(Γ) определяется с помощью полиморфизмов Γ , можно предположить, что сложность SCSP(Γ) тоже характеризуется полиморфизмами Γ . Так в [2] была доказана NP-трудность SCSP(Γ) для случаев, когда все полиморфизмы Γ существенно-унарны. Тем не менее, было показано, что одних полиморфизмов Γ недостаточно для описания сложности SCSP(Γ) [14].

Задача *Surj-Hom* вызывала интерес исследователей уже достаточно давно. Несмотря на простоту формулировки, даже для некоторых простых графов \mathcal{H} долгое время не удавалось понять, решается ли задача за полиномиальное время. Например, для цикла C_6 без петель задача была впервые отдельно сформулирована в 1999 году [16], но доказана лишь в 2017 [17]. Тем не менее, предпринималось множество попыток связать сложность задачи с различными свойствами графов. Так, в [2] было доказано, что если граф \mathcal{H} связан и содержит ровно 2 петли, то *Surj-Hom*(\mathcal{H}) является NP-трудной. В [6] было показано, что если каждая связная компонента \mathcal{H} является звездой без петель (*звезда* – это граф, который содержит ровно одну вершину, соединенную со всеми остальными вершинами графа) или полным двудольным графом без петель, то *Surj-Hom*(\mathcal{H}) лежит в P . В [7] была описана сложность *Surj-Hom* для некоторых рефлексивных ориентированных графов, и определена сложность для всех ориентированных графов на не более, чем трёх вершинах.

В 2015 году была доказана NP-трудность *Surj-Hom* для рефлексивного цикла длины 4 C_4^{ref} [10]. В [15] была описана сложность *Surj-Hom* для деревьев, содержащих как минимум одну петлю. В [13] Викас описывает алгоритм, который позволяет решить задачу за полиномиальное время для некоторых классов входных графов, таких как лес. Также в этой статье было заявлено, что существует доказательство сложности *Surj-Hom* для всех графов с не более, чем четырьмя вершинами. Это доказательство было независимо получено в [2] со ссылками на [1, 15, 10, 9].

Несмотря на большое количество результатов по данной теме, сложность *Surj-Hom* известна для очень маленького множества графов. Поэтому, в качестве первого шага в изучении этой задачи можно рассмотреть простые графы, такие как циклы. Так, для рефлексивных циклов заданной длины n C_n^{ref} сложность *Surj-Hom*(C_n^{ref}) известна только для $n \leq 4$ [10].

2. Основные результаты

Для удобства будем отождествлять граф и бинарное отношение смежности на множестве его вершин. В разделе 2.1 мы докажем что все сюръективные полиморфизмы C_n^{ref} при $n \geq 4$ существенно-унарны:

Теорема 2. Пусть $f \in \text{SPol}(C_n^{ref})$, $n \geq 4$, тогда f существенно-унарен.

Отметим, что в [18] приводится иное доказательство этого результата. После этого мы рассмотрим отношение $1IN3 = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ на множестве E_2 . Известно, что задача $\text{CSP}(1IN3)$ является NP-трудной [11]. В разделе 2.2 мы опишем сведение $\text{CSP}(1IN3)$ к $\text{SCSP}(C_n^{ref})$ для $n \geq 7$, тем самым доказав, что эта задача (и, как следствие, *Surj-Hom*(C_n^{ref})) является NP-трудной. В результате мы докажем следующую теорему, которая является переформулировкой теоремы 1.

Теорема 3. $\text{SCSP}(C_n^{ref})$ является NP-трудной при $n \geq 7$.

Наконец, в разделе 2.3 мы построим контрпримеры, которые покажут, что сконструированное в разделе 2.2 доказательство не подходит для доказательства NP-трудности задачи для циклов длины 5, 6.

2.1. Структура сюръективных полиморфизмов $\text{CSP}(C_n^{ref})$

Рассмотрим рефлексивный цикл C_n^{ref} длины n с множеством вершин $E_n = \{0, 1, \dots, n-1\}$ и отношением смежности, которое задается следующим образом

$$(x, y) \in C_n^{ref} \iff (x - y \pmod{n}) \in \{1, 0, n-1\}.$$

E_n с операцией сложения по модулю n – группа, через \oplus и \ominus будем обозначать операции сложения и вычитания в этой группе: $x \oplus y = (x + y) \bmod n$, $x \ominus y = (x - y) \bmod n$. Для $a \in E_n$ через $\ominus a$ будем обозначать элемент E_n , обратный a (так, $\ominus 1 = n - 1$). Определим на E_n метрику следующим образом:

$$\rho(x, y) = \min((y \ominus x), (x \ominus y)).$$

эта метрика соответствует расстоянию между вершинами x и y в графу C_n^{ref} . Расстояние между любыми двумя элементами E_n ограничено сверху:

Лемма 1. Для любых $x, y \in E_n$ $\rho(x, y) \leq \lfloor \frac{n}{2} \rfloor$.

Доказательство. Если $y \ominus x > \lfloor \frac{n}{2} \rfloor$, то $x \ominus y = n - (y \ominus x) \leq \lfloor \frac{n}{2} \rfloor$. \square

Заметим, что для отношения смежности C_n^{ref} верно:

$$(x, y) \in C_n^{ref} \iff \rho(x, y) \leq 1.$$

Рассмотрим множество E_n^k , образованное наборами из k элементов E_n . Элементы этого множества будем называть *векторами* и обозначать как $\bar{x} = (x_1, x_2, \dots, x_k)$. Для $\bar{x} \in E_n^k$ через x_i , $i \in \{1, \dots, k\}$ будем обозначать i -ый член \bar{x} (будем также говорить, что x_i – i -ая координата \bar{x}). Введем на этом пространстве метрику ρ : если $\bar{x}, \bar{y} \in E_n^k$, $\bar{x} = (x_1, \dots, x_k)$, $\bar{y} = (y_1, \dots, y_k)$, то

$$\rho(\bar{x}, \bar{y}) = \max_i (\rho(x_i, y_i)).$$

Для $\bar{x}, \bar{y} \in E_n^k$ через $\bar{x} + \bar{y}$, $\bar{x} - \bar{y}$ будем обозначать операции поэлементного сложения и вычитания на этом множестве. Иными словами, если $\bar{x} = (x_1, x_2, \dots, x_k)$, $\bar{y} = (y_1, y_2, \dots, y_k)$, то $\bar{x} + \bar{y} = (x_1 \oplus y_1, x_2 \oplus y_2, \dots, x_k \oplus y_k)$, $\bar{x} - \bar{y} = (x_1 \ominus y_1, x_2 \ominus y_2, \dots, x_k \ominus y_k)$.

Заметим, что k -местная функция $f : E_n^k \rightarrow E_n$ является полиморфизмом C_n^{ref} тогда и только тогда, когда $\forall \bar{x}, \bar{y} : \rho(\bar{x}, \bar{y}) \leq 1 \Rightarrow \rho(f(\bar{x}), f(\bar{y})) \leq 1$. Два вектора $\bar{x} = (x_1, x_2, \dots, x_k)$ и $\bar{y} = (y_1, y_2, \dots, y_k)$ будем называть *соседями*, если $\rho(\bar{x}, \bar{y}) \leq 1$. Два вектора \bar{x} и \bar{y} будем называть *прямыми соседями*, если они соседи и $\exists! i \rho(x_i, y_i) = 1$, то есть эти наборы отличаются ровно по одной координате. Вектор $\bar{\Delta} = (\delta_1, \delta_2, \dots, \delta_k)$ будем называть *сдвигом*, если $\forall i \delta_i \in \{\ominus 1, 0, 1\}$. Соответственно, сдвиг $\bar{\Delta}$ будем называть *прямым сдвигом*, если $\exists! i \delta_i \in \{\ominus 1, 1\}$. Два элемента \bar{x} и \bar{y} являются соседями, если они отличаются на сдвиг, и прямыми соседями, если они отличаются на прямой сдвиг.

Путь P в E_n^k – это последовательность элементов $\bar{x}^1 - \bar{x}^2 - \dots - \bar{x}^s$ такая, что любые два соседних элемента \bar{x}^i и \bar{x}^{i+1} являются соседями.

Цикл - это путь, у которого начало и конец совпадают. Длина пути - это количество элементов в данном пути минус 1. Несложно заметить, что справедлива следующая лемма:

Лемма 2. Пусть $\bar{x}, \bar{y} \in E_n^k$. Тогда существует путь из \bar{x} в \bar{y} длины $\rho(\bar{x}, \bar{y})$.

Заметим также, что так как любые два соседних элемента в пути являются соседями, то путь можно представить последовательностью сдвигов начального элемента.

Пусть $n > 2$. Введем частичную функцию $r(x, y)$, определенную на соседних элементах E_n :

$$r(x, y) = \begin{cases} \frac{1}{n}, & \text{если } x \oplus 1 = y \\ -\frac{1}{n}, & \text{если } x \ominus 1 = y \\ 0, & \text{если } x = y \end{cases}$$

Для k -местного полиморфизма $f \in \text{Pol}(C_n^{ref})$ и пути P в E_n^k индекс $c(P, f)$ пути P над f - числовая характеристика, которая определяется следующим образом: Если

$$P = \overline{x^0} - \overline{x^1} - \overline{x^2} - \dots - \overline{x^{s-1}} - \overline{x^s},$$

то

$$c(P, f) = r(f(\overline{x^0}), f(\overline{x^1})) + r(f(\overline{x^1}), f(\overline{x^2})) + \dots + r(f(\overline{x^{s-1}}), f(\overline{x^s})).$$

Заметим, что если длина P равна s , то для любого $f \in \text{Pol}(C_n^{ref})$ $|c(P, f)| \leq \frac{s}{n}$. Заметим также, что индекс обладает аддитивностью: если конец пути P совпадает с началом пути R , и путь D построен из путей P и R , то для любого $f \in \text{Pol}(C_n^{ref})$ верно $c(D, f) = c(P, f) + c(R, f)$.

Лемма 3. Для любого k -местного полиморфизма $f \in \text{Pol}(C_n^{ref})$ и произвольного пути P из \bar{x} в \bar{y} в E_n^k существует целое t такое, что $c(P, f) = \frac{f(\bar{y}) - f(\bar{x})}{n} + t$.

Доказательство. Пусть P имеет следующий вид: $P = \overline{x^0} - \overline{x^1} - \overline{x^2} - \dots - \overline{x^s}$. При каждом переходе индекс увеличивается на $\frac{1}{n}$ (если следующее значение больше предыдущего на 1), уменьшается на $\frac{1}{n}$ (если следующее значение меньше предыдущего на 1) или не изменяется. Пусть после s переходов индекс увеличился s_1 раз, уменьшился s_2 раза и не изменился s_3 раз. $s = s_1 + s_2 + s_3$, при этом по определению $c(P, f) = \frac{s_1 - s_2}{n}$. Тогда $f(\overline{x^s}) = f(\overline{x^0}) \oplus s_1 \ominus s_2$, откуда $f(\overline{x^s}) = f(\overline{x^0}) + s_1 - s_2 + n \cdot t$ для некоторого $t \in \mathbb{Z}$, и $\frac{s_1 - s_2}{n} = c(P, f) = \frac{f(\overline{x^s}) - f(\overline{x^0})}{n} + t, t \in \mathbb{Z}$. \square

Следствие 1. Пусть f – k -местный полиморфизм C_n^{ref} , $\bar{x}, \bar{y} \in E_n^k$, P_1, P_2 – пути из \bar{x} в \bar{y} . Тогда существует целое t такое, что $c(P_1, f) - c(P_2, f) = t$.

Следствие 2. Пусть f – k -местный полиморфизм C_n^{ref} , P – цикл в E_n^k . Тогда $c(P, f)$ – целое число.

Лемма 4. Пусть f – k -местный полиморфизм C_n^{ref} , $n \geq 3$, $\bar{a}, \bar{b} \in E_n^k$ такие, что $f(\bar{a}) = t$ и $f(\bar{b}) = t \oplus 1$. Тогда существует путь P из \bar{a} в \bar{b} такой, что $c(P, f) = \frac{1}{n}$.

Доказательство. По лемме 1 $\rho(\bar{x}, \bar{y}) \leq \lfloor \frac{n}{2} \rfloor$, откуда по лемме 2 существует путь P из \bar{a} в \bar{b} длины не больше, чем $\lfloor \frac{n}{2} \rfloor$. По лемме 3 $c(P, f) = \frac{1}{n} + t, t \in \mathbb{Z}$. Но $|c(P, f)| \leq \frac{\lfloor \frac{n}{2} \rfloor}{n}$, откуда при $n \geq 3$ имеем $t = 0$ и $c(P, f) = \frac{1}{n}$. \square

Лемма 5. Пусть f – сюръективный k -местный полиморфизм C_n^{ref} , $n \geq 3$. Тогда в E_n^k существует цикл C индекса 1.

Доказательство. Рассмотрим векторы $\bar{x}_0, \bar{x}_1, \dots, \bar{x}_{n-1}$ такие, что $f(\bar{x}_i) = i, i \in E_n$ (эти элементы существуют из сюръективности f). По лемме 4 существуют пути $P_{0,1}, P_{1,2}, \dots, P_{n-2,n-1}, P_{n-1,0}$ такие, что $\forall i \in E_n$ $P_{i,i \oplus 1}$ идет из \bar{x}_i в $\bar{x}_{i \oplus 1}$ и $c(P_{i,i \oplus 1}, f) = \frac{1}{n}$.

Рассмотрим путь P , последовательно собранный из путей $P_{i,i \oplus 1}$:

$$P = \bar{x}_0 - \dots - \bar{x}_1 - \dots - \dots - \bar{x}_{n-1} - \dots - \bar{x}_0.$$

Он является циклом. При этом по определению индекса пути

$$c(P, f) = c(P_{0,1}, f) + c(P_{1,2}, f) + \dots + c(P_{n-2,n-1}, f) + c(P_{n-1,0}, f) = \frac{1}{n} \cdot n = 1.$$

\square

Рассмотрим произвольный путь P в пространстве E_n^k :

$$P = \bar{x}^0 - \bar{x}^1 - \dots - \bar{x}^s.$$

Этот путь можно представить в виде последовательности сдвигов начального элемента: для $\bar{\Delta}^i = \bar{x}^i - \bar{x}^{i-1}$ имеем

$$\bar{x}^s = \bar{x}^0 + \bar{\Delta}^1 + \bar{\Delta}^2 + \dots + \bar{\Delta}^s.$$

Аналогично, произвольный элемент $\bar{x}^0 \in E_n^k$ и последовательность сдвигов $\bar{\Delta}^1, \bar{\Delta}^2, \dots, \bar{\Delta}^s$ задают путь $P = \bar{x}^0 - \bar{x}^1 - \dots - \bar{x}^s$, где $\bar{x}^i = \bar{x}^0 + \bar{\Delta}^1 + \bar{\Delta}^2 + \dots + \bar{\Delta}^i$.

Пусть путь P задается начальным элементом \bar{x}^0 и сдвигами $\bar{\Delta}^1, \bar{\Delta}^2, \dots, \bar{\Delta}^s$. Опишем ряд преобразований, которые позволяют из пути P получить новый путь R по следующим правилам:

- 1) Если в последовательности существует сдвиг $\overline{\Delta} = (e_1, \dots, e_k)$ такой, что в нем есть как минимум два ненулевых элемента $e_i, e_j \neq 0$, то строим R в котором вместо $\overline{\Delta}$ последовательно стоят два сдвига $\overline{\varepsilon}, \overline{\Delta}'$, где $\overline{\varepsilon} = (0, \dots, 0, e_i, 0, \dots, 0)$, $\overline{\Delta}' = (e_1, \dots, e_{i-1}, 0, e_{i+1}, \dots, e_k)$.
- 2) Если в последовательности существуют идущие подряд прямые сдвиги $\overline{\varepsilon}_1, \overline{\varepsilon}_2$ такие, что в $\overline{\varepsilon}_1$ ненулевой элемент стоит на i -ом месте, в $\overline{\varepsilon}_2$ ненулевой элемент стоит на j -ом месте и $i > j$, то строим R , в котором меняем эти сдвиги местами: $\overline{\varepsilon}_2, \overline{\varepsilon}_1$.
- 3) Если в последовательности существуют идущие подряд сдвиги $\overline{\Delta}_1, \overline{\Delta}_2$ такие, что $\overline{\Delta}_2 = -\overline{\Delta}_1$, то строим R , в котором удаляем эти сдвиги из цикла.

Заметим, что данные преобразования не меняют начальную и конечную точки пути, поскольку не меняют общую сумму сдвигов, образующих путь.

Докажем, что данные преобразования не меняют индекс пути.

Лемма 6. Пусть f – k -местный полиморфизм C_n^{ref} , $n \geq 4$, P – путь в E_n^k , R – путь, полученный из P применением любого из преобразований 1 – 3. Тогда $c(P, f) = c(R, f)$.

Доказательство. Рассмотрим произвольный путь $P = \overline{x^0} - \overline{x^1} - \dots - \overline{x^s}$, путь R получен из него применением любого из преобразований 1-3. R имеет такие же начальную и конечную точки, что и P , отсюда по следствию 1 индексы R и P отличаются на целое число.

Преобразование 1 добавляет один новый элемент в путь. Пусть в P к $\overline{\Delta}^i$ применяем преобразование 1: $\overline{\Delta}^i \rightarrow \overline{\varepsilon}, \overline{\Delta}', \overline{\Delta}^i = \overline{\varepsilon} + \overline{\Delta}'$. Тогда в R появляется новый элемент $\overline{x'} = \overline{x^0} + \overline{\Delta}^1 + \dots + \overline{\Delta}^{i-1} + \overline{\varepsilon}$:

$$R = \overline{x^0} - \overline{x^1} - \dots - \overline{x^{i-1}} - \overline{x'} - \overline{x^i} - \overline{x^{i+1}} - \dots - \overline{x^s}.$$

При этом преобразовании индекс меняется не более, чем на $\frac{3}{n}$: вместо $r(f(\overline{x^{i-1}}), f(\overline{x^i}))$ появляется $r(f(\overline{x^{i-1}}), f(\overline{x'})) + r(f(\overline{x'}), f(\overline{x^i}))$, и, так как $n > 3$, то индекс не меняется.

Преобразование 2 удаляет один элемент и добавляет один элемент. Пусть в P последовательно идут $\overline{\Delta}^i = \overline{\varepsilon}_1$ и $\overline{\Delta}^{i+1} = \overline{\varepsilon}_2$, в $\overline{\varepsilon}_1$ ненулевой элемент на l -ой позиции, в $\overline{\varepsilon}_2$ ненулевой элемент на m -ой позиции и $l > m$. Тогда в R вместо $\overline{x^i} = \overline{x^0} + \overline{\Delta}^1 + \dots + \overline{\Delta}^{i-1} + \overline{\varepsilon}_1$ будет $\overline{x'} = \overline{x^0} + \overline{\Delta}^1 + \dots + \overline{\Delta}^{i-1} + \overline{\varepsilon}_2$:

$$R = \overline{x^0} - \overline{x^1} - \dots - \overline{x^{i-1}} - \overline{x'} - \overline{x^{i+1}} - \dots - \overline{x^s}.$$

Заметим, что $\overline{x^i}$ и $\overline{x'}$ являются соседями, поскольку $\overline{x^i} = \overline{x^{i-1}} + \overline{\varepsilon}_1$, $\overline{x'} = \overline{x^{i-1}} + \overline{\varepsilon}_2$ и в $\overline{\varepsilon}_1$ и $\overline{\varepsilon}_2$ ненулевые элементы стоят на разных местах.

Индекс пути меняется на четыре слагаемых: вместо $r(f(\overline{x^{i-1}}), f(\overline{x^i})) + r(f(\overline{x^i}), f(\overline{x^{i+1}}))$ появляется $r(f(\overline{x^{i-1}}), f(\overline{x^i})) + r(f(\overline{x^i}), f(\overline{x^{i+1}}))$. При этом, если

$$r(f(\overline{x^{i-1}}), f(\overline{x^i})) = -r(f(\overline{x^{i-1}}), f(\overline{x^i})),$$

то

$$f(\overline{x^i}) = f(\overline{x^{i-1}}) \oplus 1, f(\overline{x^i}) = f(\overline{x^{i-1}}) \ominus 1$$

или

$$f(\overline{x^i}) = f(\overline{x^{i-1}}) \ominus 1, f(\overline{x^i}) = f(\overline{x^{i-1}}) \oplus 1,$$

откуда $f(\overline{x^i}) = f(\overline{x^{i-1}}) \ominus 2$ или $f(\overline{x^i}) = f(\overline{x^{i-1}}) \oplus 2$ – так как $n > 3$, имеем противоречие с тем, что $\overline{x^i}$ и $\overline{x^{i-1}}$ – соседи и f – полиморфизм. Отсюда индекс меняется не более, чем на $\frac{3}{n} < 1$, и преобразование 2 не меняет индекс.

Преобразование 3 убирает два элемента из пути: если в P $\overline{\Delta^i} = -\overline{\Delta^{i+1}}$, то $\overline{x^{i-1}} = \overline{x^{i+1}}$ и путь R выглядит следующим образом:

$$R = \overline{x^0} - \overline{x^1} - \dots - \overline{x^{i-1}} - \overline{x^{i+2}} - \dots - \overline{x^s}.$$

При этом индекс R отличается от индекса P на $r(f(\overline{x^{i-1}}), f(\overline{x^i})) + r(f(\overline{x^i}), f(\overline{x^{i+1}}))$, то есть не более, чем на $\frac{2}{n}$. Так как $n > 3$, то индекс пути при преобразовании 3 не меняется. \square

С помощью преобразований 1 - 3 получим цикл, состоящий исключительно из прямых сдвигов одной из координат:

Лемма 7. Для произвольного k -местного $f \in \text{SPol}(C_n^{ref})$, $n \geq 4$, существуют $\bar{a} = (a_1, \dots, a_k) \in E_n^k$, $e \in \{1, \ominus 1\}$, $i \in \{1, \dots, k\}$ такие, что $\forall b \in E_n : f(a_1, \dots, a_{i-1}, a_i \oplus b \cdot e, a_{i+1}, \dots, a_k) = b$.

Доказательство. Рассмотрим цикл C , полученный в лемме 5. Пусть этот цикл начинается в некоторой точке $\bar{c} = (c_1, \dots, c_k)$ такой, что $f(\bar{c}) = t$. Получим из него цепочку циклов $C_0 = C, C_1, C_2, \dots$ следующим образом: последовательно применяем преобразование 1, если его можно применить хотя бы к одному сдвигу, после этого последовательно применяем преобразование 2, пока существуют пары сдвигов, к которым оно применимо, после чего применяем преобразование 3, пока оно применимо. Эта цепочка конечна: для фиксированного пути P длины s в пространстве E_n^k преобразование 1 можно последовательно применить не более $s \cdot k$ раз, преобразование 2 - не более s^2 раз, преобразование 3 - не более s раз. Рассмотрим C' – последний цикл в этой цепочке. Он состоит исключительно из последовательностей одинаковых прямых сдвигов: s_1 прямых сдвигов по первой координате, s_2 прямых сдвигов по второй координате, ..., s_k прямых сдвигов по k -ой координате. Так как C' – цикл, и

каждый прямой сдвиг меняет ровно одну координату на 1 или $\ominus 1$, то число сдвигов по каждой координате кратно n . Значит, цикл C' состоит из $\frac{s_1}{n} + \frac{s_2}{n} + \dots + \frac{s_k}{n}$ циклов (n прямых сдвигов по любой координате в одном направлении дают цикл, так как $x_i \oplus n \cdot e_i = x_i$, $e_i \in \{1, \ominus 1\}$). Каждый из этих циклов имеет индекс 1, -1 или 0, при этом индекс C' равен 1, значит существует цикл C'' длины n и индекса 1, состоящий только из прямых сдвигов $\bar{\varepsilon} = (0, \dots, 0, e, 0, \dots, 0)$ (где ненулевой e стоит на i -ом месте). Из определения индекса цикла следует, что при каждом таком прямом сдвиге значение полиморфизма вырастает на один. Отсюда $f(c_1, \dots, c_k) = t$, $f(c_1, \dots, c_{i-1}, c_i \oplus e, c_{i+1}, \dots, c_k) = t \oplus 1$, $f(c_1, \dots, c_{i-1}, c_i \oplus 2e, c_{i+1}, \dots, c_k) = t \oplus 2$, ..., $f(c_1, \dots, c_{i-1}, c_i \oplus (n-1) \cdot e, c_{i+1}, \dots, c_k) = t \oplus (n-1)$. Тогда для $\bar{a} = (c_1, \dots, c_{i-1}, c_i \ominus t \cdot e, c_{i+1}, \dots, c_k)$ верно $f(\bar{a} + b \cdot \bar{\varepsilon}) = b$, $b \in E_n$. \square

Пользуясь этим, докажем теорему 2:

Теорема 2. Пусть $f \in \text{SPol}(C_n^{\text{ref}})$, $n \geq 4$, тогда f существенно-унарен.

Доказательство. По лемме 7 существуют $\bar{a} = (a_1, \dots, a_k) \in E_n^k$, $e \in \{1, \ominus 1\}$, $i \in \{1, \dots, k\}$ такие, что $\forall b \in E_n : f(a_1, \dots, a_{i-1}, a_i \oplus b \cdot e, a_{i+1}, \dots, a_k) = b$, или $f(\bar{a} + b \cdot \bar{\varepsilon}) = b$, где $\bar{\varepsilon} = (0, \dots, 0, e, 0, \dots, 0)$ – вектор, где e стоит на i -ой позиции. Без ограничения общности будем считать, что $i = k$.

Рассмотрим множество E_n^{k-1} . Через \bar{a}' будем обозначать элемент этого множества, равный (a_1, \dots, a_{k-1}) . Докажем, что для произвольного $\bar{y}' = (y_1, \dots, y_{k-1}) \in E_n^{k-1}$ и для произвольного $s \in E_n$ верно $f(y_1, \dots, y_{k-1}, a_k \oplus s \cdot e) = s$. Проведем доказательство индукцией по $\rho(\bar{y}', \bar{a}')$.

База индукции. Пусть $\rho(\bar{y}', \bar{a}') = 1$. Рассмотрим произвольное $s \in E_n$, через \bar{y} обозначим вектор $(y_1, \dots, y_{k-1}, a_k \oplus s \cdot e)$. Верно, что \bar{y} – сосед $\bar{a} + s \cdot \bar{\varepsilon}$, при этом $f(\bar{a} + s \cdot \bar{\varepsilon}) = s$, значит, $f(\bar{y})$ равен s , $s \oplus 1$ или $s \ominus 1$. Но \bar{y} также является соседом $\bar{a} + (s \oplus 1) \bar{\varepsilon}$ и $\bar{a} + (s \ominus 1) \bar{\varepsilon}$ и $f(\bar{a} + (s \oplus 1) \bar{\varepsilon}) = s \oplus 1$, $f(\bar{a} + (s \ominus 1) \bar{\varepsilon}) = s \ominus 1$. Отсюда, $f(\bar{y}) = s$.

Индуктивный переход. Пусть для всех $\bar{y}' = (y_1, \dots, y_{k-1}) \in E_n^{k-1}$ таких, что $\rho(\bar{y}', \bar{a}') \leq m$ и для всех $s \in E_n$ верно, что $f(y_1, \dots, y_{k-1}, a_k \oplus s \cdot e) = s$. Возьмем \bar{x}' такой, что $\rho(\bar{x}', \bar{a}') = m + 1$. По лемме 2 существует путь из \bar{a}' в \bar{x}' длины $m + 1$. Рассмотрим предпоследний элемент этого пути $\bar{z}' = (z_1, \dots, z_{k-1})$. Зафиксируем произвольный $s \in E_n$. $\rho(\bar{z}', \bar{y}') = m$, откуда по предположению индукции для $\bar{z} = (z_1, \dots, z_{k-1}, a_k \oplus s \cdot e)$ верно $f(\bar{z}) = s$, $f(\bar{z} \oplus \bar{\varepsilon}) = s \oplus 1$, $f(\bar{z} \ominus \bar{\varepsilon}) = s \ominus 1$. При этом вектор $\bar{x} = (x_1, \dots, x_{k-1}, a_k \oplus s \cdot e)$ является соседом \bar{z} , $\bar{z} \oplus \bar{\varepsilon}$ и $\bar{z} \ominus \bar{\varepsilon}$, откуда $f(\bar{x}) = s$.

Таким образом, мы доказали, что если f – k -местный сюръективный полиморфизм C_n^{ref} , то существуют $i \in \{1, \dots, k\}$, g – функция от одной переменной такие, что $\forall \bar{x} \in E_n^k f(\bar{x}) = g(x_i)$, то есть f – существенно-унарная функция. \square

Следствие 3. Все сюръективные полиморфизмы C_n^{ref} , $n \geq 4$, задаются следующим образом:

$$f(x_1, \dots, x_k) = e \cdot x_i + c \pmod{n},$$

где $e \in \{1, \ominus 1\}$, $c \in E_n$, $i \in \{1, \dots, k\}$.

2.2. NP-трудность SCSP(C_n^{ref})

В данном разделе будет доказана теорема 2 об NP-трудности SCSP(C_n^{ref}) для $n \geq 7$.

Рассмотрим отношение $1IN3 = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ на множестве E_2 . Задача CSP(1IN3) является NP-трудной [11].

Через π_i^j , $i \leq j$ будем обозначать j -местную проекцию на i -ую переменную, то есть $\pi_i^j(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_j) = x_i$. Для множества E через Proj_E будем обозначать множество всех проекций на E .

Сигнатурой будем называть множество функциональных символов Σ , дополненное отображением $ar : \Sigma \rightarrow \mathbb{N}$, которое сопоставляет каждому функциональному символу его арность.

Для сигнатуры Σ и множества переменных $\{x_1, \dots, x_n\}$ *минорное условие (minor condition)* \mathcal{T} над Σ – это множество тождеств вида $f(x_{i_1}, \dots, x_{i_n}) = t(x_{j_1}, \dots, x_{j_m})$, где $f, t \in \Sigma$ и $i_1, \dots, i_n, j_1, \dots, j_m \in \{1, \dots, n\}$.

Пусть M – множество функций с одинаковой областью определения. Минорное условие \mathcal{T} над Σ удовлетворяется на M , если существует отображение $F : \Sigma \rightarrow M$ такое, что:

- $\forall g \in \Sigma : g$ и $F(g)$ имеют одинаковую арность.
- Для каждого $f(x_{i_1}, \dots, x_{i_n}) = t(x_{j_1}, \dots, x_{j_m})$ из \mathcal{T} $F(f)(x_{i_1}, \dots, x_{i_n}) = F(t)(x_{j_1}, \dots, x_{j_m})$ выполняется для любой подстановки значений из области определения M вместо x_1, \dots, x_n .

Будем говорить, что отображение $F : \Sigma \rightarrow M$ является решением \mathcal{T} .

Пусть $\Sigma = \{f\}$, где f – трехместный символ. Рассмотрим систему

$$\begin{aligned} f(x, x, y) &= f(x, y, x) \\ f(x, y, x) &= f(y, x, x) \\ f(y, x, x) &= f(x, x, x). \end{aligned}$$

Это условие удовлетворяется на множестве $\{\text{MAJ}\}$, где MAJ – функция голосования от трех переменных в P_2 . При подстановке MAJ вместо f каждое тождество выполняется на всех наборах переменных из P_2 . Однако, эта система не удовлетворяется на множестве $\text{Proj}_{\{0,1\}}$: проекция

на первую переменную не может быть из-за третьего уравнения, проекция на вторую переменную – из-за второго уравнения, и проекция на третью переменную – из-за первого уравнения.

Рассмотрим произвольный экземпляр \mathcal{I} задачи CSP(1IN3):

$$\mathcal{I} = 1IN3(u_{i_1}, u_{j_1}, u_{k_1}) \wedge 1IN3(u_{i_2}, u_{j_2}, u_{k_2}) \wedge \dots \wedge 1IN3(u_{i_m}, u_{j_m}, u_{k_m}),$$

где $i_1, j_1, k_1, i_2, j_2, k_2, \dots, i_m, j_m, k_m \in \{1, 2, \dots, l\}$.

Возьмем сигнатуру $\Sigma_{\mathcal{I}} = \{f_1, \dots, f_m, g_1, \dots, g_l\}$, где f_1, \dots, f_m – трехместные символы и g_1, \dots, g_l – двуместные. Построим минорное условие $\psi(\mathcal{I})$ над $\Sigma_{\mathcal{I}}$ следующим образом:

1) Для каждого $s \in \{1, \dots, m\}$ добавим в $\psi(\mathcal{I})$ следующие тождества:

$$\begin{aligned} f_s(y, x, x) &= g_{i_s}(x, y) \\ f_s(x, y, x) &= g_{j_s}(x, y) \\ f_s(x, x, y) &= g_{k_s}(x, y) \end{aligned}$$

2) Для каждой пары $i, j \in \{1, \dots, m\}$ такой, что $i \neq j$ добавим в $\psi(\mathcal{I})$ следующие тождества:

$$f_i(x, x, x) = f_j(x, x, x).$$

В получившемся минорном условии каждому вхождению 1IN3 в \mathcal{I} будет соответствовать определенный функциональный символ f_i , а каждой переменной u_j – функциональный символ g_j .

Рассмотрим экземпляр $\mathcal{I} = 1IN3(u_1, u_2, u_3) \wedge 1IN3(u_2, u_4, u_5)$. Тогда соответствующее ему минорное условие $\psi(\mathcal{I})$ будет выглядеть следующим образом:

$$\begin{aligned} f_1(y, x, x) &= g_1(x, y) \\ f_1(x, y, x) &= g_2(x, y) \\ f_1(x, x, y) &= g_3(x, y) \\ f_2(y, x, x) &= g_2(x, y) \\ f_2(x, y, x) &= g_4(x, y) \\ f_2(x, x, y) &= g_5(x, y) \\ f_1(x, x, x) &= f_2(x, x, x). \end{aligned}$$

Более подробно конструкция минорных условий и их применение в алгебраическом анализе сложности CSP описаны в [20, 12].

Пусть \mathcal{I} – произвольный экземпляр CSP(1IN3), E – произвольное множество, $|E| > 1$. Решения \mathcal{I} и $\psi(\mathcal{I})$ связаны: \mathcal{I} имеет решение тогда и только тогда, когда $\psi(\mathcal{I})$ имеет решение на Proj_E . Эта лемма является следствием замечания 3.15 и примера 2.17 из [20].

Рассмотрим произвольное минорное условие \mathcal{T} . Покажем, что для произвольного $E, |E| > 1$, решение \mathcal{T} на множестве проекций Proj_E соответствует решению этого минорного условия на множестве функций с ровно одной существенной переменной.

Лемма 8. Пусть $S = \{f_1, f_2, \dots, f_k\}$ – сигнатура, \mathcal{T} – минорное условие над S , E, V – произвольные множества, $|E|, |V| > 1$, $n \geq 4$, \mathcal{F}_V^{un} – множество функций, определённых на V , существенно зависящих ровно от одной переменной. Тогда \mathcal{T} имеет решение на \mathcal{F}_V^{un} тогда и только тогда, когда \mathcal{T} имеет решение на Proj_E .

Доказательство. Пусть \mathcal{T} имеет решение на Proj_E . Любое множество проекций из Proj_E будет соответствовать аналогичному множеству проекций на Proj_V . Так как $|V| > 1$, то $\text{Proj}_V \subset \mathcal{F}_V^{un}$, и \mathcal{T} имеет решение на \mathcal{F}_V^{un} .

Пусть \mathcal{T} имеет решение на \mathcal{F}_V^{un} . Рассмотрим $P = \{p_1, p_2, \dots, p_l\}$ – множество функций из решения. Пусть $F : S \rightarrow P$ – решение. Каждая функция из P имеет вид

$$p_i(x_{i_1}, \dots, x_{i_{n_i}}) = g_i(x_{i_j})$$

для некоторой унарной g_i и $j \in \{1, \dots, n_i\}$, $i \in \{1, \dots, l\}$. Построим отображение $F' : S \rightarrow \text{Proj}_V$ следующим образом: если

$$F(f_i) = g_i(x_{i_j}),$$

то

$$F'(f_i) = x_{i_j}.$$

Данное отображение является решением \mathcal{T} : если при подстановке F тождество принимало вид $g(x_t) = g(x_t)$ для некоторой переменной x_t и унарной функции g , то при подстановке F' это тождество примет вид $x_t = x_t$. Отсюда, \mathcal{T} имеет решение на Proj_V , которое соответствует решению на Proj_E . \square

Следствие 4. Пусть $n \geq 4$, \mathcal{I} – произвольный экземпляр $\text{CSP}(1IN3)$. Тогда \mathcal{I} имеет решение тогда и только тогда, когда $\psi(\mathcal{I})$ имеет решение на $\text{SPol}(C_n^{ref})$.

Рассмотрим произвольный полиморфизм $f(x_1, \dots, x_k) \in \text{Pol}(C_n^{ref})$. Он определяется множеством своих значений на всех n^k наборах переменных из E_n^k . При этом $f(x_1, \dots, x_k)$ является полиморфизмом отношения $C_n^{ref} \iff \forall x_1, \dots, x_k, y_1, \dots, y_k \in E_n : \rho((x_1, \dots, x_k), (y_1, \dots, y_k)) \leq 1 \Rightarrow$

$(f(x_1, \dots, x_k), f(y_1, \dots, y_k)) \in C_n^{ref}$. Тогда, если закодировать значения полиморфизма n^k переменными $f(0, \dots, 0), \dots, f(n-1, \dots, n-1)$, то определение полиморфизма можно записать в виде формулы \mathcal{J}_f над отношением C_n^{ref} :

$$\mathcal{J}_f = \bigwedge_{\substack{x_1, \dots, x_k, y_1, \dots, y_k \in E_n: \\ \rho((x_1, \dots, x_k), (y_1, \dots, y_k)) \leq 1}} C_n^{ref}(f(x_1, \dots, x_k), f(y_1, \dots, y_k)).$$

Иными словами, необходимо рассмотреть все соседние наборы (x_1, \dots, x_k) , $(y_1, \dots, y_k) \in E_n^k$ и записать для них условие $C_n^{ref}(f(x_1, \dots, x_k), f(y_1, \dots, y_k))$.

Для произвольной сигнатуры $\Sigma = \{f_1, \dots, f_l\}$ требование того, что все f_1, \dots, f_l являются полиморфизмами C_n^{ref} можно записать в виде формулы \mathcal{J}_Σ :

$$\mathcal{J}_\Sigma = \bigwedge_{i \in \{1, \dots, l\}} \mathcal{J}_{f_i}.$$

Рассмотрим произвольное минорное условие \mathcal{T} над $\Sigma = \{f_1, \dots, f_l\}$. Построим формулу $\phi(\mathcal{T})$ над C_n^{ref} следующим образом: для каждого тождества \mathcal{T} вида $f_i(x_{i,1}, \dots, x_{i,n_i}) = f_j(x_{j,1}, \dots, x_{j,n_j})$ отождествим в \mathcal{J}_Σ значения $f_i(x_{i,1}, \dots, x_{i,n_i})$ и $f_j(x_{j,1}, \dots, x_{j,n_j})$ на всех наборах переменных $x_{i,1}, \dots, x_{i,n_i}, x_{j,1}, \dots, x_{j,n_j} \in E_n$. Набор полиморфизмов, удовлетворяющий \mathcal{T} , задает подстановку, которая по построению удовлетворяет $\phi(\mathcal{T})$. Аналогично, любое решение $\phi(\mathcal{T})$ соответствует набору полиморфизмов, который является решением \mathcal{T} . Отсюда, справедлива следующая лемма:

Лемма 9. Пусть \mathcal{T} – минорное условие. Тогда каждое решение \mathcal{T} на $\text{Pol}(C_n^{ref})$ соответствует решению $\phi(\mathcal{T})$, а каждое решение $\phi(\mathcal{T})$ соответствует решению \mathcal{T} на $\text{Pol}(C_n^{ref})$. Если \mathcal{T} имеет решение на $\text{SPol}(C_n^{ref})$, то $\phi(\mathcal{T})$ имеет сюръективное решение.

Рассмотрим \mathcal{I} – экземпляр $\text{CSP}(1\text{IN}3)$. Пусть $\psi(\mathcal{I})$ задается над сигнатурой $\Sigma_{\mathcal{I}} = \{f_1, \dots, f_k, u_1, \dots, u_l\}$, где f_1, \dots, f_k – трехместные, u_1, \dots, u_l – двуместные. Рассмотрим формулу $\phi(\psi(\mathcal{I}))$. По следствию 4 и лемме 10 если \mathcal{I} имеет решение, то $\phi(\psi(\mathcal{I}))$ имеет сюръективное решение. Докажем, что для $n \geq 7$ если $\phi(\psi(\mathcal{I}))$ имеет сюръективное решение, то $\psi(\mathcal{I})$ имеет решение на $\text{SPol}(C_n^{ref})$ (откуда по следствию 4 будет следовать, что \mathcal{I} имеет решение).

Рассмотрим сюръективное решение $\phi(\psi(\mathcal{I}))$. Каждая переменная в этом решении соответствует значению полиморфизма $p_i(x, y, z)$ для некоторых $i \leq k, x, y, z \in E_n$ (по построению каждая двуместная $p_i, i > k$ получается из некоторой трехместной $p_j, j \leq k$ путем отождествления пары переменных). По построению $\forall i, j \in \{1, \dots, k\}, x \in E_n : p_i(x, x, x) = p_j(x, x, x)$. Через $p(x, x, x)$ будем обозначать значение каждой $p_i(x, x, x)$.

Диагональю решения d будем называть количество различных значений $p(x, x, x)$ на всех $x \in E_n$, $1 \leq d \leq n$. Заметим, что если показать, что $d = n$, то это докажет, что каждый полиморфизм p_1, \dots, p_{k+l} является сюръективным.

Докажем несколько лемм, помогающих оценить d .

Лемма 10. *Для любых $x, y, z \in E_n$ существует $r \in E_n$ такой, что $\rho((x, y, z), (r, r, r)) \leq c$, $c = \lceil \frac{\lfloor \frac{2n}{3} \rfloor}{2} \rceil$.*

Доказательство. Заметим, что если в E_n есть путь длины s , то все элементы, через которые проходит этот путь (включая начальный и конечный), лежат в $\lceil \frac{s}{2} \rceil$ окрестности какого-то элемента E_n .

Возьмем произвольные $x, y, z \in E_n$. Они соответствуют вершинам x, y, z в C_n^{ref} . Между любыми двумя из выбранных вершин есть два пути: один имеет длину s и не проходит через третью вершину, другой проходит через третью вершину и имеет длину $n - s$. Можно выбрать две вершины так, чтобы путь, который не проходит через третью вершину, имел длину $s \geq \lceil \frac{n}{3} \rceil$. Отсюда, путь между этими вершинами, проходящий через третью вершину, имеет длину $s' \leq n - \lceil \frac{n}{3} \rceil = \lfloor \frac{2n}{3} \rfloor$. Значит, эти вершины лежат в $c = \lceil \frac{\lfloor \frac{2n}{3} \rfloor}{2} \rceil$ -окрестности какого-то элемента r , откуда $\rho(x, r) \leq c$, $\rho(y, r) \leq c$ и $\rho(z, r) \leq c$. \square

Лемма 11. *Пусть f – k -местный полиморфизм C_n^{ref} , $\bar{x}, \bar{y} \in E_n^k$, $f(\bar{x}) = a$, $f(\bar{y}) = b$, P – путь из \bar{x} в \bar{y} . Тогда P проходит через элементы со значениями $a \oplus 1, a \oplus 2, \dots, b \ominus 1$ или значениями $b \oplus 1, b \oplus 2, \dots, a \ominus 1$.*

Доказательство. Пусть P имеет вид

$$P = \bar{x}^0 - \bar{x}^1 - \dots - \bar{x}^s,$$

где $\bar{x}^0 = \bar{y}$, $\bar{x}^s = \bar{x}$. Рассмотрим последовательность значений элементов этого пути в f . Так как f – полиморфизм, то для любого $i \in \{0, \dots, s-1\}$ элементы $f(\bar{x}^i)$ и $f(\bar{x}^{i+1})$ являются соседями. Тогда эта последовательность образует путь G в E_n :

$$G = f(\bar{x}^0) - f(\bar{x}^1) - \dots - f(\bar{x}^s).$$

G соответствует пути в графе C_n^{ref} из вершины $f(\bar{x}^0) = a$ в вершину $f(\bar{x}^s) = b$. Этот путь проходит через вершины $a \oplus 1, a \oplus 2, \dots, b \ominus 1$ или $b \oplus 1, b \oplus 2, \dots, a \ominus 1$, откуда P также проходит через элементы с этими значениями. \square

Лемма 12. *Если произвольный $f \in \text{Pol}(C_n^{ref})$ принимает больше, чем $\lfloor \frac{n}{2} \rfloor + 1$ значений, то он сюръективен.*

Доказательство. Пусть $f \in \text{Pol}(C_n^{\text{ref}})$ принимает только m значений $s, s \oplus 1, \dots, s \oplus m \oplus 1$ (по лемме 11 f должна принимать последовательные значения). Пусть также $\lfloor \frac{n}{2} \rfloor + 1 < m < n$. Заметим, что в таком случае f не принимает $s \oplus 1, s \oplus m$. Тогда существуют \bar{x}, \bar{y} такие, что $f(\bar{x}) = s, f(\bar{y}) = s \oplus m \oplus 1$. По лемме 1 $\rho(\bar{x}, \bar{y}) \leq \lfloor \frac{n}{2} \rfloor$, значит, по лемме 2 существует путь из \bar{x} в \bar{y} длиной не более, чем $\lfloor \frac{n}{2} \rfloor$. Однако, по лемме 11 любой путь из \bar{x} в \bar{y} должен принимать все $m - 2$ промежуточных значений, отсюда, длина этого пути должна быть $m - 1$. Но $m - 1 > \lfloor \frac{n}{2} \rfloor$. Противоречие. Значит, либо $m \leq \lfloor \frac{n}{2} \rfloor + 1$, либо $m = n$. \square

Лемма 13. Пусть \mathcal{I} – произвольный экземпляр задачи CSP(1IN3). Тогда для каждого сюръективного решения $\phi(\psi(\mathcal{I}))$ диагональ этого решения $d \geq n - 2c$, где $c = \lceil \lfloor \frac{2n}{3} \rfloor \rceil$.

Доказательство. По лемме 10 любой элемент $(x, y, z) \in E_n^3$ находится на расстоянии не больше, чем c от какого-то диагонального (r, r, r) , при этом из сюръективности каждое значение $\{0, \dots, n - 1\}$ принимается в какой-то точке (x, y, z) на какой-то функции p_i . Это значит, что если на диагонали принимаются значения $\{s, s \oplus 1, \dots, s \oplus d \oplus 1\}$, то c -окрестность этих значений должна покрывать все $\{0, \dots, n - 1\}$, то есть $d + 2c \geq n$. \square

Лемма 14. Пусть \mathcal{I} – произвольный экземпляр CSP(1IN3). Для каждого сюръективного решения $\phi(\psi(\mathcal{I}))$ верно:

- Если n – чётное, то диагональ решения $d = n$ или $d < 3$.
- Если n – нечётное, то диагональ решения $d = n$ или $d < 2$.

Доказательство. Пусть n – чётное, $d \geq 3$, на диагонали принимаются значения $s, s \oplus 1, s \oplus 2$. Пусть p_1, p_2, \dots, p_k – решение, $\forall i \in \{1, \dots, k\} : p_i(r, r, r) = s \oplus 1$.

$\exists x, y, z \in E_n, j \in \{1, \dots, k\} : p_j(x, y, z) = \frac{n}{2} \oplus s \oplus 1$. По лемме 1 $\rho((x, y, z), (r, r, r)) \leq \frac{n}{2}$, откуда по лемме 2 между этими элементами есть путь длины не более, чем $\frac{n}{2}$, принимающий на p_j значения

$$s \oplus 1, s \oplus 2, \dots, \frac{n}{2} \oplus s, \frac{n}{2} \oplus s \oplus 1$$

или

$$\frac{n}{2} \oplus s \oplus 1, \frac{n}{2} \oplus s \oplus 2, \dots, s, s \oplus 1.$$

Заметим, что этот путь не принимает либо s , либо $s \oplus 2$. При этом, оба этих значения принимаются на диагонали. Тогда функция p_j всего принимает как минимум $\frac{n}{2} + 2$ значения, откуда по лемме 11 она сюръективна, и по следствию 3 $d = n$.

Пусть теперь n – нечётное, $d \geq 2$, на диагонали принимаются элементы $s, s \oplus 1$. Пусть $\forall i \in \{1, \dots, k\} : p_i(r, r, r) = s \oplus 1$. Из сюръективности $\exists x, y, z \in E_n, j \in \{1, \dots, k\} : p_j(x, y, z) = \lfloor \frac{n}{2} \rfloor \oplus s \oplus 1$. По лемме 1 $\rho((x, y, z), (r, r, r)) \leq \lfloor \frac{n}{2} \rfloor$, откуда по лемме 2 между этими элементами есть путь длины не более, чем $\lfloor \frac{n}{2} \rfloor$. Этот путь принимает на p_j значения

$$s \oplus 1, s \oplus 2, \dots, \lfloor \frac{n}{2} \rfloor \oplus s, \lfloor \frac{n}{2} \rfloor \oplus s \oplus 1$$

(и не принимает значение s), откуда всего функция p_j принимает $\lfloor \frac{n}{2} \rfloor \oplus 2$ значения, значит, по лемме 12 она сюръективна, и по следствию 3 $d = n$. \square

Теорема 4. Пусть \mathcal{I} – произвольный экземпляр CSP(1IN3), $n \geq 7$. Тогда $\psi(\mathcal{I})$ имеет решение на $\text{SPol}(C_n^{ref})$ тогда и только тогда, когда $\phi(\psi(\mathcal{I}))$ имеет сюръективное решение.

Доказательство. Пусть $\psi(\mathcal{I})$ задается над $\Sigma_{\mathcal{I}} = \{f_1, \dots, f_k, u_1, \dots, u_l\}$, где f_1, \dots, f_k – трехместные символы, u_1, \dots, u_l – двуместные.

Пусть $\psi(\mathcal{I})$ имеет решение на $\text{SPol}(C_n^{ref})$. Рассмотрим решение – множество сюръективных полиморфизмов $p_1, \dots, p_k, p_{k+1}, \dots, p_{k+l}$. Оно соответствует подстановке в переменные $\phi(\psi(\mathcal{I}))$. По построению эта подстановка будет решением, при этом, так как каждая функция p_1, \dots, p_{k+l} – сюръективная, то и подстановка в $\phi(\psi(\mathcal{I}))$ будет сюръективной.

Пусть существует сюръективное решение $\phi(\psi(\mathcal{I}))$. Оно соответствует набору полиморфизмов $p_1, \dots, p_k, p_{k+1}, \dots, p_{k+l}$. Пусть d – диагональ этого решения. Пусть также $d < n$. Тогда по лемме 14 $d < 3$ для четных n и $d < 2$ для нечётных n . Но по лемме 13 $d \geq n - 2c(n)$, где $c(n) = \lceil \lfloor \frac{2n}{3} \rfloor \rceil$. Пусть $t(n) = n - 2c(n)$. Заметим, что $t(n+6) = t(n) + 2$. При этом $t(7) = 3, t(8) = 2, t(9) = 3, t(10) = 4, t(11) = 3, t(12) = 4$. Отсюда для четных n таких, что $n \geq 10$ верно $d \geq n - 2c(n) \geq 3$, но $d < 3$ – противоречие. Аналогично, для нечетных n таких, что $n \geq 7$ верно $d \geq n - 2c(n) \geq 2$, но $d < 2$. Отсюда, для $n = 7, n \geq 9$ имеем $d = n$, каждая функция принимает на диагональных элементах все значения, значит, сюръективное решение $\phi(\psi(\mathcal{I}))$ соответствует решению $\psi(\mathcal{I})$ на $\text{SPol}(C_n^{ref})$.

Рассмотрим теперь случай $n = 8$. Из леммы 14 следует, что для любого сюръективного решения $\phi(\psi(\mathcal{I}))$ верно $d < 3$ или $d = 8$. При этом, если $d < 8$, то из леммы 13 следует, что $d \geq 2$.

Пусть $d = 2$. Тогда на диагональных элементах принимаются какие-то значения $s, s \oplus 1$. Пусть m – число различных диагональных элементов, на которых принимается значение s , $1 \leq s \leq 8$. Тогда значение $s \oplus 1$ принимается на $8 - m$ элементах. Из сюръективности решения существуют $i, j \in \{1, \dots, k\}$, $x, y, z, x', y', z' \in E_8$ такие, что $p_i(x, y, z) = s \oplus 4$ и $p_j(x', y', z') = s \oplus 5$. Расстояние от (x, y, z) до всех (r, r, r) таких, что

$p_i(r, r, r) = s$ должно быть 4. Заметим, что таких элементов может быть не более трех: это $(x \oplus 4, x \oplus 4, x \oplus 4)$, $(y \oplus 4, y \oplus 4, y \oplus 4)$ и $(z \oplus 4, z \oplus 4, z \oplus 4)$ (некоторые из них могут совпадать). Отсюда, $m \leq 3$. Аналогично, если $p_j(x', y', z') = s \oplus 5$, то значение $s \oplus 1$ может приниматься только на элементах $(x' \oplus 4, x' \oplus 4, x' \oplus 4)$, $(y' \oplus 4, y' \oplus 4, y' \oplus 4)$ и $(z' \oplus 4, z' \oplus 4, z' \oplus 4)$ откуда $8 - m \leq 3$. Противоречие. Отсюда, при $n = 8$ верно $d = n$ и сюръективное решение $\phi(\psi(\mathcal{I}))$ соответствует решению $\psi(\mathcal{I})$ на $\text{SPol}(C_n^{ref})$. \square

Наконец, докажем теорему 3 об NP-трудности $\text{SCSP}(C_n^{ref})$.

Теорема 3. $\text{SCSP}(C_n^{ref})$ является NP-трудной при $n \geq 7$.

Доказательство. Рассмотрим произвольный экземпляр $\mathcal{I} \in \text{CSP}(1IN3)$. По следствию 4 \mathcal{I} имеет решение тогда и только тогда, когда $\psi(\mathcal{I})$ имеет решение на $\text{SPol}(C_n^{ref})$, при этом по теореме 4 при $n \geq 7$ $\psi(\mathcal{I})$ имеет решение на $\text{SPol}(C_n^{ref})$ тогда и только тогда, когда существует сюръективное решение $\phi(\psi(\mathcal{I}))$. Отсюда при $n \geq 7$ произвольный экземпляр $\mathcal{I} \in \text{CSP}(1IN3)$ имеет решение тогда и только тогда, когда определенный экземпляр $\text{SCSP}(C_n^{ref})$ имеет решение. Таким образом мы свели решение $\text{CSP}(1IN3)$ к решению $\text{SCSP}(C_n^{ref})$. При этом $\text{CSP}(1IN3)$ является NP-трудной, откуда при $n \geq 7$ $\text{SCSP}(C_n^{ref})$ также является NP-трудной. \square

2.3. Контрпримеры для случаев $n = 5, 6$

В данном разделе мы продемонстрируем примеры, которые показывают, почему предложенная в разделе 2.2 конструкция не подходит для определения сложности рефлексивных циклов длины 5, 6. Так как доказательство строилось на сведении NP-трудной задачи $\text{CSP}(1IN3)$, достаточно показать такой экземпляр \mathcal{I} этой задачи, что он не имеет удовлетворяющей подстановки, тогда как формула $\phi(\psi(\mathcal{I}))$ имеет сюръективную подстановку.

Теорема 5. Пусть $n = 5, 6$. Тогда существует такой экземпляр \mathcal{I} задачи $\text{CSP}(1IN3)$, что \mathcal{I} не имеет решения, но $\phi(\psi(\mathcal{I}))$ имеет сюръективное решение.

Доказательство. Рассмотрим следующий \mathcal{I} :

$$\mathcal{I} = 1IN3(x_1, x_2, x_3) \wedge 1IN3(x_4, x_5, x_6) \wedge 1IN3(x_1, x_5, x_7).$$

Очевидно, он не имеет решения, так как $1IN3(x_1, x_1, x_1)$ либо содержит три единицы, либо не содержит единиц вообще. Минорное условие $\psi(\mathcal{I})$ определяется следующим образом:

$$\begin{aligned} f_1(y, x, x) &= g_1(x, y) & f_2(y, x, x) &= g_4(x, y) & f_3(y, x, x) &= g_1(x, y) \\ f_1(x, y, x) &= g_2(x, y) & f_2(x, y, x) &= g_5(x, y) & f_3(x, y, x) &= g_5(x, y) \\ f_1(x, x, y) &= g_3(x, y) & f_2(x, x, y) &= g_6(x, y) & f_3(x, x, y) &= g_7(x, y) \end{aligned}$$

$$\begin{aligned}f_1(x, x, x) &= f_2(x, x, x) \\f_2(x, x, x) &= f_3(x, x, x).\end{aligned}$$

Любое решение $\phi(\psi(\mathcal{I}))$ соответствует решению $\psi(\mathcal{I})$ на полиморфизмах C_n^{ref} .

Пусть $n = 5$. Рассмотрим следующие полиморфизмы:

$$p_1(x, y, z) = \begin{cases} 2, & \text{если } (x, y, z) = (2, 3, 0), \\ 1, & \text{если } \rho((x, y, z), (2, 3, 0)) = 1, \\ 0, & \text{иначе.} \end{cases}$$

$$p_2(x, y, z) = \begin{cases} 3, & \text{если } (x, y, z) = (1, 4, 2), \\ 4, & \text{если } \rho((x, y, z), (1, 4, 2)) = 1, \\ 0, & \text{иначе.} \end{cases}$$

$$p_3(x, y, z) = \begin{cases} 1, & \text{если } (x, y, z) \in \{(1, 4, 4), (2, 4, 4), (3, 4, 4)\} \\ 4, & \text{если } (x, y, z) \in \{(1, 0, 1), (1, 1, 1), (1, 2, 1), \\ & (2, 0, 2), (2, 1, 1), (2, 2, 2)\} \\ 0, & \text{иначе.} \end{cases}$$

$$\begin{aligned}p_{1_1}(x, y) &= p_1(y, x, x), & p_{2_1}(x, y) &= p_2(y, x, x), \\ p_{1_2}(x, y) &= p_1(x, y, x), & p_{2_2}(x, y) &= p_2(x, y, x), \\ p_{1_3}(x, y) &= p_1(x, x, y), & p_{2_3}(x, y) &= p_2(x, x, y), \\ h(x, y) &= 0.\end{aligned}$$

Затем, рассмотрим следующее отображение F :

$$\begin{aligned}F(f_1) &= p_1, & F(g_3) &= p_{1_3}, \\ F(f_2) &= p_2, & F(g_4) &= p_{2_1}, \\ F(f_3) &= p_3, & F(g_5) &= p_{2_2}, \\ F(g_1) &= p_{1_1}, & F(g_6) &= p_{2_3}, \\ F(g_2) &= p_{1_2}, & F(g_7) &= h.\end{aligned}$$

Можно легко убедиться в том, что это отображение действительно будет решением $\psi(\mathcal{I})$ на $\text{Pol}(C_5^{ref})$. При этом, решение $\phi(\psi(\mathcal{I}))$, соответствующее F , будет сюръективным, так как $p_1(0, 0, 0) = 0, p_1(1, 4, 0) = 1, p_2(2, 3, 0) = 2, p_2(4, 1, 2) = 3, p_2(0, 0, 1) = 4$. Значит, \mathcal{I} не имеет решения, а $\psi(\phi(\mathcal{I}))$ имеет сюръективное решение.

Пусть теперь $n = 6$. Если полиморфизмы из решения $\psi(\mathcal{I})$ не сюръективны, то, согласно леммам 14, 15, диагональ d этого решения должна быть равна 2. Рассмотрим следующие полиморфизмы:

$$p_1(x, y, z) = \begin{cases} 3, & \text{если } (x, y, z) = (0, 2, 4), \\ 2, & \text{если } \rho((x, y, z), (0, 2, 4)) = 1, \\ 0, & \text{если } (x, y, z) \in \{(1, 1, 1), (3, 3, 3), (5, 5, 5)\}, \\ 1, & \text{иначе.} \end{cases}$$

$$p_2(x, y, z) = \begin{cases} 4, & \text{если } (x, y, z) = (1, 3, 5), \\ 5, & \text{если } \rho((x, y, z), (1, 3, 5)) = 1, \\ 1, & \text{если } (x, y, z) \in \{(0, 0, 0), (2, 2, 2), (4, 4, 4)\}, \\ 0, & \text{иначе.} \end{cases}$$

$$p_3(x, y, z) = \begin{cases} 2, & \text{если } (x, y, z) \in \{(5, 3, 3), (0, 3, 3), (1, 3, 3)\} \\ 5, & \text{если } (x, y, z) \in \{(0, 2, 0), (0, 3, 0), (0, 4, 0)\} \\ 0, & \text{если } (x, y, z) \in \{(1, 1, 1), (3, 3, 3), (5, 5, 5)\}, \\ 1, & \text{иначе.} \end{cases}$$

$$\begin{aligned} p_{1_1}(x, y) &= p_1(y, x, x), & p_{2_1}(x, y) &= p_2(y, x, x), \\ p_{1_2}(x, y) &= p_1(x, y, x), & p_{2_2}(x, y) &= p_2(x, y, x), \\ p_{1_3}(x, y) &= p_1(x, x, y), & p_{2_3}(x, y) &= p_2(x, x, y), \\ & & h(x, y) &= 0. \end{aligned}$$

Затем, рассмотрим следующее отображение:

$$\begin{aligned} F(f_1) &= p_1, & F(g_3) &= p_{1_3}, \\ F(f_2) &= p_2, & F(g_4) &= p_{2_1}, \\ F(f_3) &= p_3, & F(g_5) &= p_{2_2}, \\ F(g_1) &= p_{1_1}, & F(g_6) &= p_{2_3}, \\ F(g_2) &= p_{1_2}, & F(g_7) &= h. \end{aligned}$$

Данное отображение удовлетворяет всем тождествам (так, на диагональных элементах все полиморфизмы совпадают и выводят 0 или 1). Значит, F является решением $\psi(\mathcal{I})$ на $\text{Pol}(C_6^{ref})$. Подстановка, в $\phi(\psi(\mathcal{I}))$ соответствующая этому решению, является сюръективной, так как $p_1(1, 1, 1) = 0$, $p_1(0, 0, 0) = 1$, $p_1(1, 2, 3) = 2$, $p_1(0, 2, 4) = 3$, $p_2(1, 3, 5) = 4$, $p_2(0, 2, 4) = 5$. Значит, \mathcal{I} не имеет решения, но $\phi(\psi(\mathcal{I}))$ имеет сюръективное решение. \square

Complexity of surjective homomorphism problem to reflexive cycles

Korchagin N.P.

In this article we explore the complexity of the decision problem which takes a graph as an input and asks, if there is a surjective homomorphism from it to some fixed graph \mathcal{G} . We prove, that if \mathcal{G} is a reflexive cycle with 7 or more vertices, then this problem is NP-hard.

The proof is based on a novel approach, which enables reduction of a constraint satisfaction problem to the problem of satisfying a system of equations, which, in turn, is reduced to solving surjective constraint satisfaction problem.

Keywords: surjective homomorphisms, computation complexity, constraint satisfaction, polymorphisms.

References

- [1] Bodirsky, M., Kára, J. & Martin, B. The complexity of surjective homomorphism problems—a survey. *Discrete Applied Mathematics*. **160**, 1680-1690 (2012), <https://www.sciencedirect.com/science/article/pii/S0166218X12001333>
- [2] Chen, H. An algebraic hardness criterion for surjective constraint satisfaction. *Algebra Universalis*. **72** pp. 393-401 (2014)
- [3] Golovach, P., Johnson, M., Martin, B., Paulusma, D. & Stewart, A. Surjective H-colouring : new hardness results.. *Computability*.. **8**, 27-42 (2019), <http://dro.dur.ac.uk/24228/>
- [4] Bulatov, A. A Dichotomy Theorem for Nonuniform CSPs. *2017 IEEE 58th Annual Symposium On Foundations Of Computer Science (FOCS)*. pp. 319-330 (2017), <http://arxiv.org/abs/1703.03021>
- [5] Zhuk, D. A Proof of the CSP Dichotomy Conjecture. *J. ACM*. **67** (2020,8), <https://doi.org/10.1145/3402029>
- [6] Focke, J., Goldberg, L. & Živný, S. The Complexity of Counting Surjective Homomorphisms and Compactions. *SIAM Journal On Discrete Mathematics*. **33**, 1006-1043 (2019),
- [7] Larose, B., Martin, B. & Paulusma, D. Surjective H-Colouring over Reflexive Digraphs. *ACM Trans. Comput. Theory*. **11** (2018,11), <https://doi.org/10.1145/3282431>
- [8] Vikas, N. Computational Complexity Relationship between Compaction, Vertex-Compaction, and Retraction. *Journal Of Discrete Algorithms*. **52-53** pp. 168-181 (2018), <https://www.sciencedirect.com/science/article/pii/S1570866718301631>, Combinatorial Algorithms – Special Issue Devoted to Life and Work of Mirka Miller
- [9] Vikas, N. A complete and equal computational complexity classification of compaction and retraction to all graphs with at most four vertices and some general results. *Journal Of Computer And System Sciences*. **71**, 406-439 (2005), <https://www.sciencedirect.com/science/article/pii/S0022000004000911>
- [10] Martin, B. & Paulusma, D. The computational complexity of disconnected cut and 2K2-partition. *Journal Of Combinatorial Theory, Series B*. **111** pp. 17-37 (2015), <https://www.sciencedirect.com/science/article/pii/S0095895614001026>

- [11] Schaefer, T. The Complexity of Satisfiability Problems. *Proceedings Of The Tenth Annual ACM Symposium On Theory Of Computing*. pp. 216-226 (1978), <https://doi.org/10.1145/800133.804350>
- [12] Barto, L. Algebraic Theory of Promise Constraint Satisfaction Problems, First Steps. *Fundamentals Of Computation Theory: 22nd International Symposium, FCT 2019, Copenhagen, Denmark, August 12-14, 2019, Proceedings*. pp. 3-17 (2019), https://doi.org/10.1007/978-3-030-25027-0_1
- [13] Vikas, N. Algorithms for Partition of Some Class of Graphs under Compaction and Vertex-Compaction. *Algorithmica*. **67** pp. 180 - 206 (2011)
- [14] Zhuk, D. No-Rainbow Problem and the Surjective Constraint Satisfaction Problem. *Proceedings Of The 36th Annual ACM/IEEE Symposium On Logic In Computer Science*. (2021), <https://doi.org/10.1109/LICS52264.2021.9470632>
- [15] Golovach, P., Paulusma, D. & Song, J. Computing vertex-surjective homomorphisms to partially reflexive trees. *Theoretical Computer Science*. **457** pp. 86-100 (2012), <https://www.sciencedirect.com/science/article/pii/S0304397512006561>
- [16] Vikas, N. Computational complexity of compaction to irreflexive cycles. *Journal Of Computer And System Sciences*. **68**, 473-496 (2004), <https://www.sciencedirect.com/science/article/pii/S0022000003000345>
- [17] Vikas, N. Computational Complexity of Graph Partition under Vertex-Compaction to an Irreflexive Hexagon. *42nd International Symposium On Mathematical Foundations Of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*. **83** pp. 69:1-69:14 (2017), <https://doi.org/10.4230/LIPIcs.MFCS.2017.69>
- [18] Larivière, I., Larose, B. & Pullas, D. Surjective polymorphisms of reflexive cycles. (2022)
- [19] Feder, T. & Vardi, M. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal On Computing*. **28**, 57-104 (1998),
- [20] Barto, L., Buln, J., Krokhin, A. & Opršal, J. Algebraic Approach to Promise Constraint Satisfaction. *J. ACM*. **68** (2021,7), <https://doi.org/10.1145/3457606>

Часть 3.
Математические модели

Замкнутые классы в функциональной системе рациональных функций с рациональными коэффициентами

Н. Ф. Алексиадис¹

Функциональная система представляет собой множество функций с некоторым набором операций, применяемых к этим функциям и приводящих к получению других функций из этого же множества.

Функциональные системы являются одним из основных объектов дискретной математики и математической кибернетики, поскольку они являются математическими моделями реальных и абстрактных управляющих систем.

Проблематика функциональных систем обширна. Одной из основных задач является проблема полноты, состоящая в описании таких подсистем функций, которые являются полными, т.е. из этих функций с помощью заданных операций над ними можно получить все функции.

Мы рассматриваем функциональную систему рациональных функций с рациональными коэффициентами, где в качестве операций выступают операции суперпозиции, и для этой системы исследуем задачу о замкнутых классах (структура, базис, число конечных и бесконечных замкнутых классов). Это обусловлено тем, что проблема полноты решается с помощью (максимальных) замкнутых классов.

В настоящей статье для функциональной системы рациональных функций с рациональными коэффициентами:

- 1) описаны в явном виде все конечные замкнутые классы;
- 2) найдено число всех конечных замкнутых классов, всех бесконечных замкнутых классов и всех замкнутых классов;
- 3) изучена задача о базисах замкнутых классов, а именно, установлено, что существует замкнутый класс, имеющий конечный базис, существует замкнутый класс, имеющий бесконечный базис, и существует замкнутый класс, не имеющий базиса; приведены конкретные примеры соответствующих замкнутых классов;

¹ Алексиадис Никос Филиппович — доцент кафедры прикладной математики и искусственного интеллекта, доцент кафедры математического и компьютерного моделирования Института информационных и вычислительных технологий Национального исследовательского университета «МЭИ», e-mail: aleksiadis@yandex.ru.

Aleksiadis N. Ph. — associate professor, National Research University «MPEI», Institute of Information and Computation Technologies, Chair of Applied Mathematics and Artificial Intelligence, Chair of Mathematical and Computer Modeling

- 4) найдено число замкнутых классов, имеющих конечный базис, число замкнутых классов, имеющих бесконечный базис, и число замкнутых классов, не имеющих базиса.

Ключевые слова: функциональная система, проблема полноты, полная система, замкнутый класс, рациональная функция.

1. Введение

Эта статья является расширенной версией моего доклада о замкнутых классах рациональных функций с рациональными коэффициентами, сделанного в 2021 году на XII Международной конференции «Интеллектуальные системы и компьютерные науки», посвященной 85-летию со дня рождения профессора В.Б. Кудрявцева [1], и ее можно считать продолжением моих статей [2] – [4] о проблеме полноты для полиномиальных и рациональных функций.

Несмотря на то, что мы используем стандартную терминологию дискретной математики, в частности, теории функциональных систем (см. [5] и [6]), с целью корректного понимания изложенного, все-таки следует уточнить некоторые моменты.

Функциональная система представляет собой множество функций с некоторым набором операций, применяемых к этим функциям и приводящих к получению других функций из этого же множества, т.е. функциональная система (ф.с.) \mathbf{F} – это пара вида $\mathbf{F} = (F, O)$, где F – множество функций, а O множество операций над функциями из F , при этом каждая операция из O замкнута относительно множества F .

Для произвольного подмножества A множества F обозначим через $[A]$ множество всех функций из F , которые получаются из функций множества A с помощью конечного числа применения операций из O . Множество $[A]$ называется *замыканием множества A* .

Множество A ($A \subseteq F$) называется *замкнутым* в функциональной системе \mathbf{F} , если $[A] = A$.

Замкнутое множество принято называть *замкнутым классом*.

Множество A ($A \subseteq F$) называется *полным* в функциональной системе \mathbf{F} , если $[A] = F$.

Полное множество принято называть *полной системой*.

Полная система называется *базисом* в функциональной системе \mathbf{F} , если никакая ее собственная подсистема не является полной в \mathbf{F} , т.е. базис – это минимальная полная система в ф.с. \mathbf{F} .

Аналогично определяются полная система и базис любого замкнутого класса в \mathbf{F} .

Проблематика теории функциональных систем обширна. Одной из центральных проблем является *проблема полноты, состоящая в описа-*

нии всех подмножеств A множества функций F , которые являются полными в ф.с. $\mathbf{F} = (F, O)$, т.е. $[A] = F$.

Как известно, изучение проблемы полноты осуществлялось путем исследования конкретных функциональных систем. В этих функциональных системах решение проблемы полноты было сведено к описанию всех предполных классов (максимальных замкнутых классов). Метод решения проблемы полноты в терминах предполных классов стал после этого одним из основных (можно сказать, стал традицией).

Следовательно, при исследовании проблемы полноты одной из основных задач является задача о замкнутых классах.

В настоящей работе решается эта задача для функциональной системы рациональных функций с рациональными коэффициентами, которая играет ключевую роль не только в самой дискретной математике и математической кибернетике, но и во многих других областях математики, например, в теории функций (аппроксимационные теоремы Чебышева и Вейерштрасса), в вычислительной математике и технике (построение и анализ вычислительных чипов и нейронных сетей). Актуальность полученных результатов также состоит и в развитии самой теории функциональных систем как в плане охвата новых модельных объектов типа рациональных функций, так и в вычленении позитивных результатов, а также в отсечении негативных ситуаций.

Введем несколько стандартных обозначений, необходимых для дальнейшего изложения.

N — множества всех натуральных чисел (включая 0),

Q — множества всех рациональных чисел,

$|A|$ — мощность множества A .

c_0 — мощность счетного множества.

$c = 2^{c_0}$ — мощность континуум.

Для удобства изложения полагаем, что $0^0 = 1$.

2. Функциональная система рациональных функций с рациональными коэффициентами

Определим функциональную систему рациональных функций с рациональными коэффициентами.

Функция вида

$$f(x_1, \dots, x_n) = \frac{g(x_1, \dots, x_n)}{h(x_1, \dots, x_n)},$$

где $g(x_1, \dots, x_n)$ и $h(x_1, \dots, x_n)$ — полиномы с рациональными коэффициентами, называется *рациональной функцией с рациональными коэффициентами*.

Рациональные функции с рациональными коэффициентами будем называть также *rq-функциями*. Обозначим через FRQ множество всех рациональных функций с рациональными коэффициентами.

Говорят, что *rq-функция* $f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$ *существенно зависит от переменной* x_i , если существуют такие два набора

$$(c_1, \dots, c_{i-1}, a, c_{i+1}, \dots, c_n) \text{ и } (c_1, \dots, c_{i-1}, b, c_{i+1}, \dots, c_n)$$

значений переменных $x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n$, что

$$f(c_1, \dots, c_{i-1}, a, c_{i+1}, \dots, c_n) \neq f(c_1, \dots, c_{i-1}, b, c_{i+1}, \dots, c_n).$$

В этом случае мы говорим, что x_i является *существенной переменной* функции $f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$.

Если x_i не является существенной переменной функции

$$f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n),$$

то она называется *фиктивной переменной* данной функции.

Пусть

$$f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \text{ и } g(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

две *rq-функции* и пусть x_i фиктивная переменная функции

$$f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n).$$

Если для любых $c_1, \dots, c_{i-1}, c_i, c_{i+1}, \dots, c_n$ значений переменных

$$x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n$$

имеем

$$f(c_1, \dots, c_{i-1}, c_i, c_{i+1}, \dots, c_n) = g(c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_n),$$

то говорят, что

$$g(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \text{ получается из } f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$$

удалением фиктивной переменной x_i и, наоборот,

$$f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \text{ получается из } g(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

добавлением фиктивной переменной x_i .

Две функции f (с областью определения $D(f)$ и областью значений $E(f)$) и g (с областью определения $D(g)$ и областью значений $E(g)$) назовем равными, если

1. f и g зависят от одинакового числа существенных переменных,
2. $D(f) = D(g)$, $E(f) = E(g)$ и $f(x) = g(x)$ для любого $x \in D(f)$.

В дальнейшем будем считать, что вместе с функцией f заданы и все равные ей функции, т.е. функции рассматриваем с точностью до фиктивных переменных.

Если дана конечная система функций f_1, f_2, \dots, f_m (где $m \geq 1$), то можно считать, что все они зависят от одних и тех же переменных x_1, x_2, \dots, x_n , т.е. имеют вид

$$f_1(x_1, x_2, \dots, x_n), f_2(x_1, x_2, \dots, x_n), \dots, f_m(x_1, x_2, \dots, x_n).$$

Если дана функция, отличная от константы, то путем отождествления переменных из нее можно получить равную ей функцию, все переменные которой являются существенными.

Функциональная система рациональных функций с рациональными коэффициентами \mathbf{F}_{RQ} – это пара $\mathbf{F}_{RQ} = (F_{RQ}, O)$, где F_{RQ} – множество всех рациональных функций с рациональными коэффициентами, а O – множество операций суперпозиции. Операции суперпозиции включают в себя:

- 1) *перестановку переменных*;
- 2) *переименование переменных (без отождествления)*;
- 3) *отождествление переменных*;
- 4) *введение фиктивной переменной*;
- 5) *удаление фиктивной переменной*;
- 6) *подстановку одной функции в другую*.

Следует отметить, что это определение функциональной системы $\mathbf{F}_{RQ} = (F_{RQ}, O)$ корректное, так как любая суперпозиция функций из F_{RQ} является опять функцией из F_{RQ} .

3. Основные результаты

Следующие две теоремы дают исчерпывающий ответ о структуре и числе конечных замкнутых классов в ф.с. \mathbf{F}_{RQ} .

Теорема 1. *В ф.с. \mathbf{F}_{RQ} существуют только следующие конечные замкнутые классы:*

- i) C , где C – произвольное конечное подмножество множества Q ;
- ii) $I_1 = \{x\}$, $I_2 = \{x; -x\}$;
- iii) $C \cup I_1, \{\pm c_1, \dots, \pm c_k\} \cup I_2$, где $\pm c_1, \dots, \pm c_k \in Q$, а C, I_1 и I_2 определяются соответственно в предыдущих пунктах.

Доказательство. Очевидно, что каждое из перечисленных множеств является конечным замкнутым классом в \mathbf{F}_{RQ} . Покажем, что в \mathbf{F}_{RQ} не существует других конечных замкнутых классов, отличных от перечисленных. Для этого достаточно доказать, что если замкнутый класс содержит rq -функцию $f(x_1, \dots, x_n)$, отличную от константы, $g(x) = x$ и $h(x) = -x$, то он содержит бесконечное число попарно различных rq -функций.

Пусть A – произвольный замкнутый класс, который содержит такую rq -функцию $f = f(x_1, \dots, x_n)$. Поскольку $f(x_1, \dots, x_n)$ отлична от константы, то она имеет существенную переменную; не ограничивая общность, можно считать, что существенными переменными функции $f(x_1, \dots, x_n)$ являются переменные x_1, \dots, x_n ($n \geq 1$) (см. замечание (2)).

Далее, возможны два случая:

1. $n = 1$, т.е. f имеет одну существенную переменную. Можно считать, что $f = f(x)$.

Так как функция f отлична от $g(x) = x$ и $h(x) = -x$, то очевидно, что последовательность функций

$$f(x), f(f(x)), f(f(f(x))), \dots$$

состоит из попарно различных функций и все они принадлежат множеству A (поскольку A – замкнутый класс).

2. $n \geq 2$, т.е. $f(x_1, \dots, x_n)$ имеет более одной существенной переменной. Без ограничения общности можно считать, что функция $f(x_1, \dots, x_n)$ существенно зависит от всех своих переменных x_1, \dots, x_n (см. замечание (3)).

Тогда очевидно, что если в функции $f(x_1, \dots, x_n)$ на место переменной x_1 подставим функцию $f(y_1, \dots, y_n)$ (где $\{y_1, \dots, y_n\} \cap \{x_1, \dots, x_n\} = \emptyset$), то получим функцию

$$f(f(y_1, \dots, y_n), x_2, \dots, x_n),$$

существенно зависящую от всех своих $2n - 1$ переменных; затем, если в функции

$$f(f(y_1, \dots, y_n), x_2, \dots, x_n)$$

на место переменной y_1 подставим функцию $f(z_1, \dots, z_n)$, где

$$\{z_1, \dots, z_n\} \cap \{y_1, \dots, y_n, x_1, \dots, x_n\} = \emptyset,$$

то получим функцию

$$f(f(f(z_1, \dots, z_n), y_2, \dots, y_n), x_2, \dots, x_n),$$

существенно зависящую от всех своих $3n - 2$ переменных и т.д.

Итак, имеем бесконечную последовательность функций

$$f(x_1, \dots, x_n), f(f(y_1, \dots, y_n), x_2, \dots, x_n), f(f(f(z_1, \dots, z_n), y_2, \dots, y_n), x_2, \dots, x_n), \dots,$$

которая состоит из попарно различных функций (поскольку числа существенных переменных этих функций попарно различны) и все они принадлежат множеству A (т.к. A – замкнутый класс). Следовательно, A содержит бесконечное число попарно различных функций. \square

Теорема 2. В функциональной системе \mathbf{F}_{RQ}

- i) число всех конечных замкнутых классов равно c_0 ;
- ii) число всех бесконечных замкнутых классов равно c ;
- iii) число всех замкнутых классов равно c .

Доказательство. 1) Справедливость первого утверждения непосредственно следует из теоремы (1), в которой перечислены все замкнутые классы в ф.с. \mathbf{F}_{PR} .

2) Понятно, что чтобы доказать мощность некоторого множества A равна $|A| = c$, достаточно показать, что $|A| \geq c$ и $|A| \leq c$.

С одной стороны, поскольку любое бесконечное подмножество

$$G = \{c_1, c_2, c_3, \dots\}$$

множества рациональных чисел Q является замкнутым классом (легко проверить, что $[G] = G$), то число всех бесконечных замкнутых классов не меньше c , так как мощность множества всех рациональных чисел равна c_0 , а, как известно, мощность множества всех бесконечных подмножеств множества мощности c_0 равна c . Но, с другой стороны, это число не может быть больше c , так как $\mathbf{F}_{RQ} = (F_{RQ}, O)$ является функциональной системой мощности c_0 , т.е. $|F_{RQ}| = c_0$ и, следовательно, число всех подмножеств множества $|F_{RQ}|$ равно c .

3) Ясно, что (число всех замкнутых классов) = (число всех конечных замкнутых классов) + (число всех бесконечных замкнутых классов), т.е. (число всех замкнутых классов) = $c_0 + c = c$. \square

А что касается базисов замкнутых классов, то оказывается, что имеют место все три случая.

Теорема 3. В функциональной системе \mathbf{F}_{RQ}

- i) существует замкнутый класс, имеющий конечный базис;
- ii) существует замкнутый класс, имеющий бесконечный базис;
- iii) существует замкнутый класс, не имеющий базиса.

Доказательство. Чтобы убедиться в этом, достаточно привести примеры соответствующих замкнутых классов.

1. Пусть $A = \{2x, 4x, 8x, \dots, 2^n x, \dots\}$, где n – любое положительное целое число. Ясно, что множество A является замкнутым классом в \mathbf{F}_{RQ} , а система $B = \{2x\}$ – его базисом. Следовательно, A замкнутый класс, имеющий конечный базис.

2. Пусть $A = \{x, 2x, 3x, \dots, mx, \dots\}$, где m – любое положительное целое число. Ясно, что множество A является замкнутым классом в \mathbf{F}_{RQ} , а система $B = \{x, 2x, 3x, 5x, \dots, px, \dots\}$, где p – любое простое число, является его базисом. Следовательно, A замкнутый класс, имеющий бесконечный базис.

3. Пусть $A = [T]$, где $T = \{1, x_1^2, x_1^2 x_2^2, x_1^2 x_2^2 x_3^2, \dots\}$. Покажем, что замкнутый класс A не имеет базиса.

Очевидно, что A состоит из всех rq -функции вида 1 и $x_1^{2k_1} \dots x_n^{2k_n}$, где n – любое положительное целое число, а k_1, \dots, k_n – любые натуральные числа.

Допустим, что класс A имеет базис; обозначим его через B .

Ясно, что B содержит константу 1 . Действительно, если из B удалить константу 1 , то полученная подсистема будет состоять из таких функций, которые сохраняют константу 0 . Из этих функций с помощью операций суперпозиции можно получить только функции, которые опять сохраняют константу 0 , т.е. нельзя получить константу 1 .

Очевидно, что B содержит еще и функцию, отличную от константы 1 ; обозначим ее через $f_1(x_1, \dots, x_{n_1})$, где $n_1 > 0$, т.е. B содержит функцию, имеющую существенную переменную. Не ограничивая общность, можно считать, что существенными переменными функции $f_1(x_1, \dots, x_{n_1})$ являются переменные x_1, \dots, x_{n_1} (см. замечание (2)).

Пусть число переменных, которые содержит эта функция во 2-ой степени, равно m_1 ($0 \leq m_1 \leq n_1$); не ограничивая общность, можно считать, что этими переменными являются переменные x_1, \dots, x_{m_1} . Следовательно, $f_1(x_1, \dots, x_{n_1})$ имеет вид

$$f_1(x_1, \dots, x_{n_1}) = x_1^2 \dots x_{m_1}^2 x_{m_1+1}^{2k_{m_1+1}} \dots x_{n_1}^{2k_{n_1}},$$

где $k_{m_1+1}, \dots, k_{n_1}$ – некоторые натуральные числа отличные, от 0 и 1 .

Очевидно, что из функций 1 и $f_1(x_1, \dots, x_{n_1})$ с помощью операций суперпозиции невозможно получить функцию

$$g(x_1, \dots, x_{m_1+1}) \equiv x_1^2 \dots x_{m_1}^2 x_{m_1+1}^2.$$

Следовательно, B содержит функцию вида

$$f_2(x_1, \dots, x_{n_2}) = x_1^2 \dots x_{m_1}^2 x_{m_1+1}^2 \dots x_{m_2}^2 x_{m_2+1}^{2l_{m_2+1}} \dots x_{n_2}^{2l_{n_2}},$$

где n_2, m_2 – некоторые положительные целые числа, при этом $n_2 > n_1$ и $n_1 < m_2 \leq n_2$, а $l_{m_2+1}, \dots, l_{n_2}$ – некоторые натуральные числа, отличные от 0 и 1 .

Очевидно, что из функций $1, f_1(x_1, \dots, x_{n_1})$ и $f_2(x_1, \dots, x_{n_2})$ с помощью операций суперпозиции невозможно получить функцию $g(x_1, \dots, x_{m_2+1}) \equiv x_1^2 \dots x_{m_2}^2 x_{m_2+1}^2$.

Следовательно, B содержит функцию вида

$$f_3(x_1, \dots, x_{n_3}) = x_1^2 \dots x_{m_2}^2 x_{m_2+1}^2 \dots x_{m_3}^2 x_{m_3+1}^{2s_{m_2+1}} \dots x_{n_3}^{2s_{n_3}},$$

где n_3, m_3 – некоторые положительные целые числа, при этом $n_3 > n_2$ и $n_2 < m_3 \leq n_3$, а $s_{m_3+1}, \dots, s_{n_3}$ – некоторые натуральные числа, отличные от 0 и 1 и т.д.

Итак, B содержит бесконечное число попарно различных функций

$$1, f_1(x_1, \dots, x_{n_1}), f_2(x_1, \dots, x_{n_2}), f_3(x_1, \dots, x_{n_3}), \dots$$

Но, с другой стороны, если рассмотрим любую бесконечную подпоследовательность этой последовательности, содержащую константу 1, то очевидно, что из функций этой подпоследовательности с помощью операции суперпозиции можно получить все функции множества T , следовательно, и все функции замкнутого класса. Следовательно, собственная подсистема базиса B является полной в ф.с. \mathbf{F}_{PR} . Получили противоречие. Итак, замкнутый класс A не имеет базиса. \square

На вопрос сколько конечных замкнутых классов, имеющих конечные базисы, имеющих бесконечные базисы или вообще не имеющих базисов, ответ дает следующая теорема.

Теорема 4. *В функциональной системе \mathbf{F}_{RQ}*

- i) число замкнутых классов, имеющих конечный базис, равно c_0 ;*
- ii) число замкнутых классов, имеющих бесконечный базис, равно c ;*
- iii) число всех замкнутых классов, не имеющих базиса, равно c .*

Доказательство. 1) Любое конечное множество вида $G = \{c_1, c_2, \dots, c_n\}$, где $c_1, c_2, \dots, c_n \in Q$, является базисом самого этого множества G ; поэтому число замкнутых классов, имеющих конечный базис, не меньше $|Q| = c_0$. Но это число не может быть больше c_0 , так как $\mathbf{F}_{RQ} = (F_{RQ}, O)$ является функциональной системой мощности c_0 , т.е. $|F_{RQ}| = c_0$ и, следовательно, число всех его конечных подмножеств множества $|F_{RQ}|$ равно c_0 .

2) Любое бесконечное множество вида

$$G = \{c_1, c_2, c_3, \dots\}, \text{ где } c_1, c_2, c_3, \dots \in Q,$$

является базисом самого этого множества G ; поэтому число замкнутых классов, имеющих бесконечный базис, не меньше c . Но это число не может быть больше c , так как $\mathbf{F}_{RQ} = (F_{RQ}, O)$ является функциональной системой мощности c_0 , т.е. $|F_{RQ}| = c_0$ и, следовательно, число всех его бесконечных подмножеств множества $|F_{RQ}|$ равно c .

3) Чтобы убедиться в справедливости последнего пункта, достаточно заметить, что множество

$$A_T = [\{1, x_1^2, x_1^2 x_2^2, x_1^2 x_2^2 x_3^2, \dots\} \cup T],$$

где T – произвольное бесконечное подмножество множества всех простых чисел, не имеет базиса и, если $T_1 \neq T_2$, где T_1, T_2 – любые подмножества множества всех простых чисел, то $A_{T_1} \neq A_{T_2}$. \square

И, наконец, рассмотрим некоторые конкретные замкнутые классы в ф.с. \mathbf{F}_{RQ} , которые играют ключевую роль при решении проблемы полноты, так как они являются предполными классами.

Подмножество A множества F_{RQ} называется *предполным* (максимальным) *классом* в \mathbf{F}_{RQ} , если $[A] \neq F_{RQ}$, но для любой функции f из $F_{RQ} \setminus A$ выполнено $[A \cup \{f\}] = F_{RQ}$.

Очевидно, что *предполный класс является замкнутым классом*.

Отметим, что для доказательства предполноты в \mathbf{F}_{RQ} множества $A (A \subseteq F_{RQ})$ нужно показать, что

- $A \neq \emptyset$ и $A \neq F_{RQ}$;
- A замкнутый класс;
- для любого f из $F_{RQ} \setminus A$ система $A \cup \{f\}$ полна в F_{RQ} .

Если функция $f(x_1, \dots, x_n)$ не определена в точке (c_1, \dots, c_n) , то будем писать $f(c_1, \dots, c_n) = *$ (следует отметить, что это просто обозначение, а не доопределение в точках разрывах).

Пусть A – произвольное подмножество множества Q . Обозначим через $V(A)$ множество всех таких функций $f(x_1, \dots, x_n)$ из F_{RQ} , что для любых значений $c_1, \dots, c_n \in A$ переменных x_1, \dots, x_n либо $f(c_1, \dots, c_n) = *$ либо $f(c_1, \dots, c_n) \in A$. В этом случае $f(x_1, \dots, x_n)$ называется *функцией, сохраняющей множество A* , а $V(A)$ – *классом, сохраняющим множество констант A* .

На интересующий нас вопрос – является ли предполным классом множество всех rq -функций, сохраняющих конечное множество констант, ответ дает следующая теорема.

Теорема 5. *Если A – произвольное (непустое) конечное подмножество множества Q , то класс $V(A)$ является предполным классом в ф.с. \mathbf{F}_{RQ} .*

Доказательство. Пусть $A = \{c_1, \dots, c_m\}$, где c_1, \dots, c_m ($m \geq 1$) – константы из Q .

Ясно, что $V(A) \neq \emptyset, V(A) \neq F_{RQ}$ и $[V(A)] = V(A)$.

Далее, пусть $f(x_1, \dots, x_n) \in F_{RQ} \setminus A$; тогда для некоторых констант c_{i_1}, \dots, c_{i_n} из A и c из $Q \setminus A$ будет выполнено $f(c_{i_1}, \dots, c_{i_n}) = c$ (и это понятно, так как если для всех констант c_{i_1}, \dots, c_{i_n} из A имеем $f(c_{i_1}, \dots, c_{i_n}) = *$ или $f(c_1, \dots, c_n) \in A$, то тогда получается, что функция $f(x_1, \dots, x_n)$ сохраняет множество A , что противоречиво).

Рассмотрим rq -функции

$$g_1(t, x, y) = \frac{(t - c_1) \cdot \dots \cdot (t - c_m)}{(c - c_1) \cdot \dots \cdot (c - c_m)} \cdot (x - y - c_1) + c_1,$$

$$g_2(t, x, y) = \frac{(t - c_1) \cdot \dots \cdot (t - c_m)}{(c - c_1) \cdot \dots \cdot (c - c_m)} \cdot (xy - c_1) + c_1,$$

$$g_3(t, x, y) = \frac{(t - c_1) \cdot \dots \cdot (t - c_m)}{(c - c_1) \cdot \dots \cdot (c - c_m)} \cdot \left(\frac{x}{y} - c_1\right) + c_1 \quad (y \neq 0),$$

$$g_4(t) = \frac{(t - c_1) \cdot \dots \cdot (t - c_m)}{(c - c_1) \cdot \dots \cdot (c - c_m)} \cdot (1 - c_1) + c_1.$$

Очевидно, что

$$g_1(t, x, y), g_2(t, x, y), g_3(t, x, y), g_4(t) \in V(A).$$

Следовательно, суперпозиции rq -функций

$$g_1(c, x, y) = x - y, \quad g_2(c, x, y) = xy, \quad g_3(c, x, y) = \frac{x}{y}, \quad g_4 = 1$$

принадлежат классу $[A \cup \{f(x_1, \dots, x_n)\}]$, т.е. $[A \cup \{f(x_1, \dots, x_n)\}]$ содержит подсистему

$$B = \left\{1, x - y, xy, \frac{x}{y}\right\},$$

которая является полной в \mathbf{F}_Q [2]. Значит, $[A \cup \{f(x_1, \dots, x_n)\}] = F_{RQ}$. \square

4. Заключение

В заключении вкратце отметим, что цель, поставленная в начале статьи, достигнута: изучена структура замкнутых классов и их базисов; найдено число замкнутых классов и их базисов для каждого возможного типа, более того, приведены соответствующие конкретные примеры для каждого случая.

Список литературы

- [1] Алексиадис Н. Ф., “О замкнутых классах в функциональной системе рациональных функций с рациональными коэффициентами”, Интеллектуальные системы. Теория и приложения. Т.25, вып. 4, 2021, 62–65.
- [2] Алексиадис Н. Ф., “Рациональные А-функции с рациональными коэффициентами”, Чебышевский сборник. т. 23, вып. 4. <https://doi.org/10.22405/2226-8383-2022-23-4-11-19>, 2022, 11–19.
- [3] Алексиадис Н. Ф., “О проблеме полноты рациональных функций с рациональными коэффициентами”, Международная конференция «Мальцевские чтения», 2021, 143.
- [4] Алексиадис Н. Ф., “О базисах рациональных функций с рациональными коэффициентами”, Материалы XX Международной конференции "Алгебра, теория чисел, дискретная геометрия и многомасштабное моделирование: современные проблемы, приложения и проблемы истории посвященной 130-летию со дня рождения академика И. М. Виноградова, 2021, 80-83.
- [5] Кудрявцев В. Б., *Функциональные системы*, Изд-во МГУ, Москва, 1982, 157 с.
- [6] Яблонский С. В., *Введение в дискретную математику*, Изд-во Наука, Москва, 1986, 384 с.

Closed classes in the functional system of rational functions with rational coefficients Aleksiadis N.Ph.

A functional system is a set of functions endowed with a set of operations on these functions. The operations allow one to obtain new functions from the existing ones.

Functional systems are mathematical models of real and abstract control systems and thus are one of the main objects of discrete mathematics and mathematical cybernetic.

The problems in the area of functional systems are extensive. One of the main problems is deciding completeness; this problem consists in the description of all subsets of functions that are complete, i.e. generate the whole set.

In our paper we consider the functional system of rational functions with rational coefficients endowed with the superposition operation for this system we study the problem of closed classes (structure, basis, number of finite and infinite closed classes).

Importance of the problem of closed classes is ensured by the fact that completeness problem can frequently be solved with the help of (maximal) closed classes.

The main results concerning the functional system of rational functions with rational coefficients presented in our paper are the following:

- 1) *all finite closed classes are described explicitly;*
- 2) *the number of finite closed classes, infinite closed classes and all closed classes is found;*
- 3) *the problem of bases of closed classes is studied, namely, it is established that there exist closed classes with a finite basis, there exist closed classes with an infinite basis, and there exist closed classes without a basis; explicit examples of the corresponding closed classes are given;*
- 4) *the number of closed classes with a finite basis, the number of closed classes with an infinite basis and the number of closed classes without a basis are established.*

Keywords: functional system, completeness problem, complete system, closed classes, rational function

References

- [1] Aleksiadis N. Ph., “On the closed classes in the functional system of rational functions”, *Intelligent Systems. Theory and Applications*, **25:4** (2021), 62–65 (In Russian).
- [2] Aleksiadis N. Ph., “Rational A-functions with rational coefficients”, *Chebyshevskii sbornik*, **23:4** (2022), 11–19. <https://doi.org/10.22405/2226-8383-2022-23-4-11-19> (In Russian).
- [3] Aleksiadis N. Ph., “On problem of completeness for rational functions with rational coefficients”, *Proc. Int. Conf. “Maltsev Readings“*, 2021, 143 (In Russian).
- [4] Aleksiadis N. Ph., “On the bases of rational functions with rational coefficients”, *Proc. 20th Int. Conf. “Algebra, number theory and discrete geometry: modern problems, applications and problems of history“*, 2021, 80-83 (In Russian).
- [5] Kudryavtsev V. B., “*Functional systems* “, Publishing House of Mekhmat. fac. MSU, Moscow (In Russian), 1982, 157 c.
- [6] Yablonsky S. V., “*Introduction to discrete mathematics“*, Science, Moscow (In Russian), 1986, 384 c.

О выразимости автоматов с операцией суперпозиции

Д. Н. Бабин¹, А. А. Летуновский²

Задача выразимости для константных автоматов и линейных автоматов для расширенной суперпозиции алгоритмически разрешима. Имеет место теорема об алгоритмической разрешимости задачи выразимости автоматов с линейными переходами над полем конечной характеристики.

Ключевые слова: расширенная суперпозиция, выразимость, линейные автоматы, алгоритмическая разрешимость.

Задача выразимости конечных автоматов относительно суперпозиции в общем случае алгоритмически неразрешима. Известная теорема Крона-Роудза [1] работает лишь в случае автоматов, имеющих полную систему переходов. Это когда всякая подстановка состояний, порождаемая словом, порождается также одной буквой. Все случаи алгоритмической неразрешимости выразимости автоматов, как правило, доказывались как неразрешимость выразимости автоматов с безусловными переходами (константных автоматов) [2].

И для этих константных автоматов Летуновскому А.А удалось решить задачу выразимости при условии наличия в выражающей конечной системе всех автоматов с одним состоянием (функций без памяти) и автомата «задержки» [3, 4]. Суперпозицию при наличии указанной добавки автор назвал «расширенной суперпозицией». При этом множество длин периодов константных автоматов, выразимых через конечную систему автоматов Σ вместе со «штрихом Шеффера» и «задержкой», представляло геометрическую прогрессию

$$(\Sigma) = \{b, bq, bq^2, \dots\}$$

, где b и q натуральные числа. Автор назвал $q = q(\Sigma)$ главным цикловым индексом системы автоматов Σ , а $b = b(\Sigma)$ безусловным цикловым индексом системы автоматов Σ . Цикловые индексы стали важным параметром для определения возможностей систем автоматов. В качестве необходимого условия выразимости конечной системы Σ_2 через конечную систему Σ_1 (при наличии указанной добавки) имеем $(\Sigma_1) \in (\Sigma_2)$.

¹Бабин Дмитрий Николаевич — МГУ им. М.И. Ломоносова, мех-мат ф-т, каф. МаТИС, профессор, e-mail: d.n.babin@mail.ru

Babin Dmitry Nikolaevich — professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

²Летуновский Алексей Александрович — e-mail: alekseyletunovskiy@gmail.com
Letunovskiy Alexey Alexandrovich.

Пусть Λ – множество автоматов над $E_k = \{0, 1, \dots, k-1\}$ с линейными переходами над конечным полем F характеристики p . Это Λ не замкнутое относительно суперпозиции множество автоматов, однако и «функция Вебба» и «задержка» также имеют линейные переходы. Представляет интерес найти цикловые индексы для системы Λ . Известно [5], что всякий автомат с линейными переходами над конечным полем F разложим в суперпозицию автомата с безусловными переходами, «задержки» Z и линейного автомата сдвигов W , переходы которого, задаются формулой $s' = s + x$. (Здесь имеется ввиду сложение в поле F). Главный цикловой индекс системы $\{Z, W, \}$ равен p . Заметим, что всякий константный автомат является автоматом с линейными переходами и безусловный цикловой индекс может быть любым натуральным числом. Таким образом, бесконечное множество автоматов с линейными переходами имеет конечный главный цикловой индекс.

Имеет место

Теорема Пусть Λ – множество автоматов над $E_k = \{0, 1, \dots, k-1\}$ $k = p^t$ с линейными переходами над конечным полем F характеристики p . Тогда существует алгоритм проверки выразимости автомата с линейными переходами для расширенной суперпозиции через конечную систему автоматов.

Список литературы

- [1] Алешин С.В., “Об одном следствии теоремы Крона-Роудза”, *Дискретная математика*, **11:4** (1999), 101–109.
- [2] Кратко М.И., “Алгоритмическая неразрешимость проблемы распознавания полноты для конечных автоматов”, *ДАН СССР*, **155:1** (1964), 35–37.
- [3] Бабин Д.Н., Летуновский А.А., “О возможностях суперпозиции, при наличии в базисе автоматов фиксированной добавки из булевых функций и задержки”, *Интеллектуальные системы*, **19:3** (2015), 71–77.
- [4] Летуновский А.А., “выразимость линейных автоматов относительно расширенной суперпозиции.”, *Интеллектуальные системы*, **19:1** (2015), 161–170.
- [5] Бабин Д.Н., “Автоматы с линейными переходами.”, *Интеллектуальные системы. Теория и приложения*, **23:3** (2019), 87–95.

About expressibility of automata with superposition Babin D.N, Letunovskiy A.A.

Expressibility of constant and linear automata for extended superposition is decidable. There is a theorem about decidability of automaton with linear transition expressibility. **Key words:** extended superposition, expressibility, linear automata, decidability.

References

- [1] Aleshin S.V., “About one consequence of Krohn-Rhodes theorem”, *Discrete Math*, **11**:4 (1999), 101–109.
- [2] Kratko M.I., “Algorithmic undecidability of completeness problem for finite automata”, *DAN USSR*, **155**:1 (1964), 35–37.
- [3] Babin D.N., Letunovskiy A.A., “About possible properties of automata superposition for basis including all boolean functions and delay automaton.”, *Intelligent systems*, **19**:3 (2015), 71–77.
- [4] Letunovskiy A.A., “Expressibility of linear aotomata for extended superposition.”, *Intelligent systems*, **19**:1 (2015), 161–170.
- [5] Babin D.N., “Automata with linear transitions”, *Intelligent systems, Theory and application.*, **23**:3 (2019), 87–95.

Об изменении длины минимальной склейки при алфавитных неисправностях

П. С. Дергач¹, Н. С. Ботирова²

Целью данной статьи является исследование характера изменения минимальной длины склейки для алфавитного кодирования при различных типах неисправностей в схемах. Рассматриваются три вида операций: удаление, добавление и замена одной буквы. Основной вопрос, изучаемый в работе, заключается в оценке того, во сколько раз может измениться длина минимальной склейки после выполнения каждой из указанных операций. В результате исследования был найден критерий сохранения свойства неоднозначности в терминах схемы кодирования, а также получены верхние и нижние оценки на скорость изменения длины минимальной склейки в каждом из трех случаев. Данные оценки являются важным практическим инструментом для проектирования алфавитных кодеров с учетом возможных неисправностей в схемах.

Ключевые слова: алфавитное кодирование, минимальная склейка, схема кодирования, алфавитное декодирование.

Введение

Данная статья продолжает исследование в области теории алфавитного кодирования. Подробнее об этом направлении можно прочитать в работах [1-3]. В данной статье рассмотрена задача об изменении длины минимальной склейки при алфавитных неисправностях. Конкретнее, речь идет о характере изменения минимальной длины склейки для алфавитного кодирования в случае, когда в элементарных кодах схемы кодирования происходят операции добавления, удаления или замены одной буквы. В статье представлены результаты исследования данной задачи, получен критерий сохранения свойства неоднозначности в терминах схемы кодирования, а также приведены оценки сверху и снизу на относительную скорость изменения минимальной длины склейки для таких

¹Дергач Пётр Сергеевич — к.ф.-м.н., м.н.с. каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: dergachpes@gmail.com.

Dergach Peter Sergeevich — Ph.D., junior researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

²Ботирова Наргизахон Санжар кизи — выпускник Филиала МГУ имени М. В. Ломоносова в городе Ташкенте, e-mail: nargiza07082002@gmail.com.

Botirova Nargizakhon Sanjar qizi — Graduate of the M. V. Lomonosov Moscow State University Branch in Tashkent.

неисправностей. Также задача исследована для трех случаев длины минимальной склейки – сумма длин прообразов, максимум длины прообраза, длина образа.

Основные определения и результаты

Большая часть определений из этого раздела взята из [4-6].

Пусть A – непустой конечный алфавит. Его элементы называем *буквами*. Конечные последовательности букв называем *словами*. *Длиной слова α* называем количество букв в этом слове с учетом кратности и обозначаем его $|\alpha|$. Множество всех слов в алфавите A обозначаем через A^* .

Пусть $\alpha = \beta^k$, где β^k – k раз повторенное слово β . Тогда называем β *измельчением* слова α . Пару слов α_1, α_2 называем *соизмеримой*, если у них есть общее измельчение.

Зафиксируем два конечных непустых алфавита A и B . Пусть есть какое-то отображение $f : A \rightarrow B^* \setminus \{\lambda\}$:

$$f(\alpha_1) = \beta_1,$$

$$f(\alpha_2) = \beta_2,$$

...

$$f(\alpha_r) = \beta_r.$$

Это отображение называется *схемой алфавитного кодирования из алфавита A в алфавит B* . Слова β_i называем *элементарными кодами*. Множество всех схем алфавитного кодирования из алфавита A в алфавит B обозначаем $F(A, B)$.

Пусть f – схема алфавитного кодирования. Доопределим отображение \tilde{f} до отображения $\tilde{f} : A^* \rightarrow B^*$ следующим образом:

$$\tilde{f}(A) = A,$$

$$\tilde{f}(\alpha_{i_1}\alpha_{i_2}\dots\alpha_{i_n}) = \beta_{i_1}\beta_{i_2}\dots\beta_{i_n}.$$

Отображение \tilde{f} называем *алфавитным кодированием из алфавита A в алфавите B* .

Склейкой для функции алфавитного кодирования \tilde{f} , порожденной схемой f , называем пару (α, β) различных слов $\alpha, \beta \in A^*$ такую, что $\tilde{f}(\alpha) = \tilde{f}(\beta)$. Множество всех склеек для f обозначаем через $U(f)$.

Пусть \tilde{f} – функция алфавитного кодирования. Для произвольной склейки $M = (\alpha, \beta) \in U(f)$ обозначаем через $l_1(M)$ сумму длин слов α, β , через $l_2(M)$ – максимум из этих длин, а через $l_3(M)$ – длину общего кода.

Минимальной склейкой первого типа для функции алфавитного кодирования \tilde{f} называем склейку M с минимальным значением $l_1(M)$. Соответствующее значение называем размером первого типа и обозначаем его через $T_1(\tilde{f})$.

Минимальной склейкой второго типа для функции алфавитного кодирования \tilde{f} называем склейку M с минимальным значением $l_2(M)$. Соответствующее значение называем размером второго типа и обозначаем его через $T_2(\tilde{f})$.

Минимальной склейкой третьего типа для функции алфавитного кодирования \tilde{f} называем склейку M с минимальным значением $l_3(M)$. Соответствующее значение называем размером третьего типа и обозначаем его через $T_3(\tilde{f})$.

Пусть f_1 и f_2 – схемы алфавитного кодирования. Будем говорить, что f_2 получена из f_1 *путем единичной неисправности первого типа*, если f_2 образована из f_1 разовой заменой одной буквы в одном элементарном коде.

Пусть f_1 и f_2 – схемы алфавитного кодирования. Будем говорить, что f_2 получена из f_1 *путем единичной неисправности второго типа*, если f_2 образована из f_1 разовым добавлением одной буквы в одном элементарном коде.

Пусть f_1 и f_2 – схемы алфавитного кодирования. Будем говорить, что f_2 получена из f_1 *путем единичной неисправности третьего типа*, если f_2 образована из f_1 разовым удалением одной буквы в одном элементарном коде.

Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Называем такую схему *монотипной*, если $f(a)$ и $f(b)$ состоят только из нулей или только из единиц.

Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Пусть $f_1 \in F(A, B)$ получено из f единичной неисправностью любого из трех типов. Если при этом $U(f)$ и $U(f_1)$ непустые, то говорим, что *данная единичная неисправность сохранит свойство однозначности*.

Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Пусть $f_1 \in F(A, B)$ получено из f единичной неисправностью типа i , где $i \in \{1, 2, 3\}$. Если при этом $U(f)$ и $U(f_1)$ непустые, то *скоростью изменения длины склейки при данном типе неисправности* называем значение $\frac{T_i(\tilde{f}_1)}{T_i(\tilde{f})}$.

Пусть $A = \{a, b\}$, $B = \{0, 1\}$, $f \in F(A, B)$ и $n \in \mathbf{N}$, причем $|f(a)|, |f(b)| \leq n$. Класс таких схем обозначаем через $F_n(A, B)$.

Утверждение 1. Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Тогда для f не существует единичных неисправностей первого типа, сохраняющих неоднозначность.

Утверждение 2. Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Тогда для единичных неисправностей второго и третьего типов свойство

сохранения неоднозначности возможно тогда и только тогда, когда f монотипна.

Утверждение 3. Пусть $A = \{a, b\}$, $B = \{0, 1\}$, $f \in F(A, B)$. И пусть схема f_1 получена из f единичной неисправностью второго типа, причем $U(f)$ и $U(f_1)$ непустые. Тогда верны следующие оценки:

$$\frac{1}{2}T_1(\tilde{f}) \leq T_1(\tilde{f}_1) \leq 2T_1(\tilde{f}),$$

$$\frac{1}{2}T_2(\tilde{f}) \leq T_2(\tilde{f}_1) \leq 2T_2(\tilde{f}),$$

$$\frac{1}{2}T_3(\tilde{f}) < T_3(\tilde{f}_1) \leq \frac{5}{2}T_3(\tilde{f}).$$

Утверждение 4. Пусть $A = \{a, b\}$, $B = \{0, 1\}$, $f \in F(A, B)$. И пусть схема f_1 получена из f единичной неисправностью третьего типа, причем $U(f)$ и $U(f_1)$ непустые. Тогда верны следующие оценки:

$$\frac{1}{2}T_1(\tilde{f}) \leq T_1(\tilde{f}_1) \leq 2T_1(\tilde{f}),$$

$$\frac{1}{2}T_2(\tilde{f}) \leq T_2(\tilde{f}_1) \leq 2T_2(\tilde{f}),$$

$$\frac{2}{5}T_3(\tilde{f}) \leq T_3(\tilde{f}_1) < 2T_3(\tilde{f}).$$

Вспомогательные утверждения

Лемма 1. Пусть $A = (a, b)$ и B – конечный алфавит. Тогда для произвольной схемы алфавитного кодирования $f \in F(A, B)$ множество $U(f)$ непусто если и только если $f(a)$ и $f(b)$ соизмеримы.

Доказательство. Пусть $f(a) = \alpha^k$ и $f(b) = \alpha^m$, тогда $\tilde{f}(ab) = \tilde{f}(ba)$. Значит $U(f)$ непусто. Обратное, пусть $U(f)$ непусто. Докажем индукцией по сумме длин k элементарных кодов $f(a)$ и $f(b)$, что они соизмеримы. База индукции $k = 2$. Тогда $f(a)$ и $f(b)$ состоят из одной буквы и очевидно, что склейка возможна тогда и только тогда, когда $f(a) = f(b)$. Переход индукции $k \rightarrow k + 1$. Пусть $(\alpha_1, \alpha_2) \in U(f)$, то есть выполнено $\tilde{f}(\alpha_1) = \tilde{f}(\alpha_2)$. Будем считать, что первые буквы α_1, α_2 различны, так как иначе их можно было бы выкинуть, сохранив свойство склейки. Без ограничения общности, α_1 начинается на a и α_2 начинается на b и длина $f(a)$ не превосходит длины $f(b)$. Тогда $f(a)$ является собственным началом $f(b)$, то есть $f(b) = f(a)\beta$ для некоторого $\beta \in A^*$. Тогда мы можем

представить $f(\alpha_1)$ и $f(\alpha_2)$ как последовательности слов $f(a)$ и β . Эти последовательности по прежнему различны, так как одна из них начинается на $f(a)\beta$, а другая – на $f(a)f(a)$. Таким образом, мы построили склейку для кодирования $f_1(a) = f(a)$, $f_1(b) = \beta$. По предположению индукции получаем, что $f(a)$ и β соизмеримы. Значит соизмеримы и $f(a), f(b)$. Переход доказан.

Лемма 2. Если слова v_1^k и v_2^m соизмеримы, то соизмеримы будут и слова v_1, v_2 .

Доказательство. Рассмотрим схему алфавитного кодирования, в которой $f(a) = v_1$ и $f(b) = v_2$. Так как выполнена цепочка равенств $\tilde{f}(a^k b^m) = v_1^k v_2^m = v_2^m v_1^k = \tilde{f}(b^m a^k)$, то по Лемме 1 получаем, что v_1 и v_2 соизмеримы. Лемма 2 доказана.

Лемма 3. Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$ – монотипная склейка, в которой $f(a) = 0^k$, $f(b) = 0^m$. Тогда $T_3(\tilde{f})$ или равно $\max(k, m)$ или равно $k + m$.

Доказательство. Без ограничения общности, считаем, что $k \geq m$. Если k делится на m , то очевидно, что минимальным образом склейки будет являться слово 0^k . Если же k не делится на m нацело, то докажем, что минимальным образом склейки будет слово 0^{k+m} . В самом деле, возьмем произвольную склейку вида $\tilde{f}(\alpha) = \tilde{f}(\beta)$, $\alpha \neq \beta$. Если хоть одно из слов α, β содержит в себе хотя бы по одной букве a и хотя бы по одной букве b , то утверждение очевидно. В противном случае, оба этих слова состоят только из a или только из b , причем одно из них – из a , а другое – из b . Но чтобы длина склейки была меньше $k + m$, букву a можно взять только один раз, что невозможно, когда k не делится на m нацело. Осталось заметить, что $\tilde{f}(ab) = \tilde{f}(ba) = 0^{k+m}$. Лемма доказана.

Доказательство основных утверждений

Утверждение 1. Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Тогда для f не существует едичных неисправностей первого типа, сохраняющих неоднозначность.

Доказательство. Докажем утверждение от противного. Пусть есть едичная неисправность первого типа, сохраняющая неоднозначность. Тогда в силу Леммы 1 $f(a)$ и $f(b)$ соизмеримы, то есть $f(a) = \nu^k$ и $f(b) = \nu^m$. Без ограничения общности, будем считать, что замена произошла со словом $f(a)$. Тогда новое слово $f'(a)$, опять же в силу Леммы 1, будет соизмеримо с $f(b)$. Но в силу Леммы 2 отсюда получаем, что $f(a)$ и $f'(a)$ соизмеримы. Это могло бы быть возможно только, если $\nu = 0$ или $\nu = 1$. Но даже в этом случае замена нарушает свойство соизмеримости. Утверждение доказано.

Утверждение 2. Пусть $A = \{a, b\}$, $B = \{0, 1\}$ и $f \in F(A, B)$. Тогда для едичных неисправностей второго и третьего типов свойство сохранения неоднозначности возможно тогда и только тогда, когда f монотипна.

Доказательство. Пусть f – монотипна, то есть $f(a) = 0^k$ и $f(b) = 0^m$. Тогда удаление или добавление 0 в $f(a)$ очевидно сохраняет свойство неоднозначности.

Обратно, пусть при добавлении буквы в схему f сохранилось свойство неоднозначности. По Лемме 1 получаем, что в этой схеме слова $f(a)$ и $f(b)$ соизмеримы. Без ограничения общности, будем считать, что буква добавлена в слово $f(a)$ и получилось слово $f'(a)$. Тогда опять же по Лемме 1, $f'(a)$ и $f(b)$ соизмеримы. Тогда по Лемме 2 соизмеримы и слова $f(a)$, $f'(a)$. Но это возможно только если $f(a) = 0^k$ (или $f(a) = 1^k$). Значит f – монотипна.

Для удаления буквы утверждение доказывается аналогично.

Утверждение доказано.

Утверждение 3. Пусть $A = \{a, b\}$, $B = \{0, 1\}$, $f \in F(A, B)$. И пусть схема f_1 получена из f единичной неисправностью второго типа, причем $U(f)$ и $U(f_1)$ непустые. Тогда верны следующие оценки:

$$\frac{1}{2}T_1(\tilde{f}) \leq T_1(\tilde{f}_1) \leq 2T_1(\tilde{f}),$$

$$\frac{1}{2}T_2(\tilde{f}) \leq T_2(\tilde{f}_1) \leq 2T_2(\tilde{f}),$$

$$\frac{1}{2}T_3(\tilde{f}) < T_3(\tilde{f}_1) \leq \frac{5}{2}T_3(\tilde{f}).$$

Доказательство. В силу Утверждения 2 сохранение свойства неоднозначности при единичной несправности второго типа возможно только если f монотипна, то есть без ограничения общности, $f(a) = 0^k$ и $f(b) = 0^m$. Без ограничения общности, считаем, что $k \geq m$.

Разберемся сначала с оценкой T_1 , то есть случаем, когда длина склейки – сумма длин входных слов. После единичной неисправности второго типа в одном из слов $f(a)$ и $f(b)$ необходимо добавить 0. При этом новая схема f_1 обязательно содержит склейку $\tilde{f}_1(ab) = \tilde{f}_1(ba)$ длины 4. Длина исходной склейки не меньше 2. Поэтому длина минимальной склейки после неисправности не могла вырасти более чем в 2 раза. С другой стороны, исходная схема f тоже имела склейку $\tilde{f}(ab) = \tilde{f}(ba)$ длины 4. Значит длина минимальной склейки для этой схемы не больше 4. А новая схема будет иметь склейку длины не меньше чем 2. Поэтому длина новой склейки не могла уменьшиться более чем в 2 раза.

Теперь докажем утверждение для оценки T_2 , то есть случая, когда длина склейки – максимум из длин входных слов. Абсолютно аналогично, после едичной неисправности второго типа в одном из слов $f(a)$ и

$f(b)$ необходимо добавить 0. При этом новая схема f_1 обязательно содержит склейку $\tilde{f}_1(ab) = \tilde{f}_1(ba)$ длины 2. Длина исходной склейки не меньше 1. Поэтому длина минимальной склейки после неисправности не могла вырасти более чем в 2 раза. С другой стороны, исходная схема f тоже имела склейку $\tilde{f}(ab) = \tilde{f}(ba)$ длины 2. Значит длина минимальной склейки для этой схемы не больше 2. А новая схема будет иметь склейку длины не меньше чем 1. Поэтому длина новой склейки не могла уменьшиться более чем в 2 раза.

Переходим к доказательству утверждения для оценки T_3 , то есть случая, когда длина склейки - длина общего кода. Абсолютно аналогично, после единичной неисправности второго типа в одном из слов $f(a)$ и $f(b)$ необходимо добавить 0. Разберем сначала случай, когда k делится на m нацело. Тогда по Лемме 3 $T_3(f) = k$ и $T_3(\tilde{f}_1)$ может быть равно или $k+1$, или k , или $k+m+1$. В самом худшем случае длина склейки могла вырасти не более чем в $\frac{k+m+1}{k} \leq \frac{2k+1}{k}$ раз. При $k=1$ получаем $f(a) = f(b) = 0$ и $T_3(\tilde{f}_1) = 2$. Если же $k > 1$, то $\frac{2k+1}{k} \leq \frac{5}{2}$. Кроме того, очевидно, что $T_3(\tilde{f}_1) \geq k = T_3(\tilde{f})$ и значит в любом случае длина склейки уменьшиться не может. Разберем теперь случай, когда k не делится нацело m . Тогда по Лемме 3 $T_3(f) = k+m$ и $T_3(\tilde{f}_1)$ может быть равно или $k+1$, или k , или $k+m+1$. В самом худшем случае длина склейки могла вырасти не более чем в $\frac{k+m+1}{k+m} \leq \frac{4}{3}$ раз, так как случай $k=m=1$ не подходит. Аналогично, в самом худшем случае длина склейки могла уменьшиться не более чем в $\frac{k+m}{k} \leq \frac{2k-1}{k} < 2$ раз.

Утверждение доказано.

Замечание 1. Оценки из Утверждения 3 в общем случае не улучшаемы. В самом деле, верхняя оценка для T_1 достигается для случая, когда $f(a) = 00$, $f(b) = 00$, $f_1(a) = 00$, $f_1(b) = 000$. Нижняя оценка для T_1 достигается для случая, когда $f(a) = 000$, $f(b) = 00$, $f_1(a) = 000$, $f_1(b) = 000$. Верхняя оценка для T_2 достигается для случая, когда $f(a) = 0$, $f(b) = 0$, $f_1(a) = 00$, $f_1(b) = 0$. Нижняя оценка для T_2 достигается для случая, когда $f(a) = 00$, $f(b) = 0$, $f_1(a) = 0$, $f_1(b) = 0$. Верхняя оценка для T_3 достигается для случая, когда $f(a) = 00$, $f(b) = 00$, $f_1(a) = 000$, $f_1(b) = 00$. Нижняя оценка для T_3 не достигается, но получается в пределе, когда $f(a) = 0^k$, $f(b) = 0^{k-1}$, $f_1(a) = 0^k$, $f_1(b) = 0^k$, $k \rightarrow \infty$.

Утверждение 4. Пусть $A = \{a, b\}$, $B = \{0, 1\}$, $f \in F(A, B)$. И пусть схема f_1 получена из f единичной неисправностью третьего типа, причем $U(f)$ и $U(f_1)$ непустые. Тогда верны следующие оценки:

$$\frac{1}{2}T_1(\tilde{f}) \leq T_1(\tilde{f}_1) \leq 2T_1(\tilde{f}),$$

$$\frac{1}{2}T_2(\tilde{f}) \leq T_2(\tilde{f}_1) \leq 2T_2(\tilde{f}),$$

$$\frac{2}{5}T_3(\tilde{f}) \leq T_3(\tilde{f}_1) < 2T_3(\tilde{f}).$$

Доказательство. В силу Утверждения 2 сохранение свойства неоднозначности при единичной несправности второго типа возможно только если f – монотипна, то есть без ограничения общности, $f(a) = 0^k$ и $f(b) = 0^m$. Без ограничения общности, считаем, что $k \geq m$.

Разберемся сначала с оценкой T_1 , то есть случаем, когда длина склейки – сумма длин входных слов. После единичной несправности третьего типа в одном из слов $f(a)$ и $f(b)$ необходимо удалить 0. При этом новая схема f_1 обязательно содержит склейку $\tilde{f}_1(ab) = \tilde{f}_1(ba)$ длины 4. Длина исходной склейки не меньше 2. Поэтому длина минимальной склейки после несправности не могла вырасти более чем в 2 раза. С другой стороны, исходная схема f тоже имела склейку $\tilde{f}(ab) = \tilde{f}(ba)$ длины 4. Значит длина минимальной склейки для этой схемы не больше 4. А новая схема будет иметь склейку длины не меньше чем 2. Поэтому длина новой склейки не могла уменьшиться более чем в 2 раза.

Теперь докажем утверждение для оценки T_2 , то есть случая, когда длина склейки – максимум из длин входных слов. Абсолютно аналогично, после единичной несправности третьего типа в одном из слов $f(a)$ и $f(b)$ необходимо удалить 0. При этом новая схема f_1 обязательно содержит склейку $\tilde{f}_1(ab) = \tilde{f}_1(ba)$ длины 2. Длина исходной склейки не меньше 1. Поэтому длина минимальной склейки после несправности не могла вырасти более чем в 2 раза. С другой стороны, исходная схема f тоже имела склейку $\tilde{f}(ab) = \tilde{f}(ba)$ длины 2. Значит длина минимальной склейки для этой схемы не больше 2. А новая схема будет иметь склейку длины не меньше чем 1. Поэтому длина новой склейки не могла уменьшиться более чем в 2 раза.

Переходим к доказательству утверждения для оценки T_3 , то есть случая, когда длина склейки – длина общего кода. Абсолютно аналогично, после единичной несправности третьего типа в одном из слов $f(a)$ и $f(b)$ необходимо удалить 0. Разберем сначала случай, когда k делится на m нацело. Тогда по Лемме 3 $T_3(f) = k$ и $T_3(\tilde{f}_1)$ может быть равно или $k - 1$, или k , или $k + m - 1$. В самом худшем случае длина склейки могла вырасти не более чем в $\frac{k+m-1}{k} \leq \frac{2k-1}{k} < 2$ раз. Кроме того, очевидно, что $T_3(\tilde{f}_1) \geq k - 1$, причем $k > 1$, так как нам необходимо иметь возможность удалить букву. Поэтому в самом худшем случае длина минимальной склейки могла уменьшиться не больше чем в $\frac{k}{k-1} \leq 2$ раз. Разберем теперь случай, когда k не делится нацело m . Тогда по Лемме 3 $T_3(f) = k + m$ и $T_3(\tilde{f}_1)$ может быть равно или $k - 1$, или k , или $k + m - 1$. В самом худшем случае длина склейки могла вырасти не более чем в $\frac{k+m-1}{k+m} < 1$ раз. Аналогично, в самом худшем случае длина склейки могла уменьшиться не более чем в $\frac{k+m}{k-1}$ раз. Заметим, что $k \geq 3$, так как

иначе $m = 1$ или $m = 2$ и k делилось бы на m . Кроме того, $m \leq k - 1$. Тогда $\frac{k+m-1}{k-1} \leq \frac{2k-1}{k-1} = 2 + \frac{1}{k-1} \leq \frac{5}{2}$.

Утверждение доказано.

Замечание 2. Оценки из Утверждения 4 в общем случае не улучшаемы. В самом деле, верхняя оценка для T_1 достигается для случая, когда $f(a) = 000$, $f(b) = 000$, $f_1(a) = 000$, $f_1(b) = 00$. Нижняя оценка для T_1 достигается для случая, когда $f(a) = 00$, $f(b) = 000$, $f_1(a) = 00$, $f_1(b) = 00$. Верхняя оценка для T_2 достигается для случая, когда $f(a) = 0$, $f(b) = 0$, $f_1(a) = 00$, $f_1(b) = 0$. Нижняя оценка для T_2 достигается для случая, когда $f(a) = 00$, $f(b) = 0$, $f_1(a) = 0$, $f_1(b) = 0$. Верхняя оценка для T_3 не достигается, но получается в пределе для случая, когда $f(a) = 0^k$, $f(b) = 0^k$, $f_1(a) = 0^{k-1}$, $f_1(b) = 0^k$, $k \rightarrow \infty$. Нижняя оценка для T_3 достигается для случая, когда $f(a) = 000$, $f(b) = 00$, $f_1(a) = 00$, $f_1(b) = 00$.

Заключение

В статье рассматривается и успешно решается задача об исследовании изменения длины минимальной склейки при единичных неисправностях в алфавитном кодировании. Рассмотрены три вида неисправностей – неисправность замены, неисправность удаления, неисправность добавления. Также рассмотрены три вида длин склеек – сумма длин прообразов, максимум длин прообразов, длина образа. В каждом из этих случаев получены точные неулучшаемые оценки на скорость изменения длины склейки.

Список литературы

- 1) С. В. Яблонский. *Введение в дискретную математику*. М.: Наука, 1986.
- 2) Ал. А. Марков. *Введение в теорию кодирования*. М: Наука, 1982.
- 3) Л. П. Жильцова. *Современные проблемы теории кодирования*. Учебное пособие, Нижний Новгород, 2007.
- 4) П. С. Дергач. *Алфавитное кодирование регулярных языков с полиномиальной функцией роста*. Кандидатская диссертация по специальности 01.01.09. 2016.
- 5) П. С. Дергач. *О проблеме вложения допустимых классов*. Интеллектуальные системы, изд. МГУ, М., том 19, № 2, с. 143-174, 2015.

- 6) П. С. Дергач. *Об однозначности алфавитного декодирования*. Дискретная математика -М.: Наука, том 24, № 4, с. 80-90, 2012.

On changing the minimum glue length for alphabetic faults **Dergach P.S., Botirova N.S**

The purpose of this thesis is to study the nature of the change in the minimum length of gluing for alphabetic coding for various types of faults in circuits. Three types of operations are considered: deletion, addition and replacement of one letter. The main issue studied in the work is to estimate how many times the length of the minimum gluing can change after performing each of the indicated operations. As a result of the study, a criterion for preserving the ambiguity property in terms of the coding scheme was found, and upper and lower bounds were obtained for the rate of change in the length of the minimum gluing in each of the three cases. These estimates are an important practical tool for designing alphabetic encodings, taking into account possible faults in circuits.

Keywords: alphabetic coding, minimum gluing, coding scheme, alphabetic encoding.

References

- [1] Yablonskiy S.V., “Introduction to discrete math”, 1986.
- [2] A.A. Markov, “Introduction to coding theory”, 1982.
- [3] Zhiltsova, “Modern problems of coding theory”, 2007.
- [4] Dergach P.S., “Alphabetic encoding of regular languages with polynomial growth function”, *PhD thesis*, 2016, 1–213.
- [5] Dergach P.S., “On the problem of embedding admissible classes”, *Intelligent Systems. Theory and Applications*, 2015, 143–174.
- [6] Dergach P.S., “On uniqueness of alphabetic decoding”, 2012, 80–90.

К сведению авторов публикаций в журнале «Интеллектуальные системы. Теория и приложения»

В соответствии с требованиями ВАК РФ к изданиям, входящим в перечень ведущих рецензируемых научных журналов и изданий, в которых могут быть опубликованы основные научные результаты диссертаций на соискание ученой степени доктора и кандидата наук, статьи в журнал «Интеллектуальные системы. Теория и приложения» предоставляются авторами в следующей форме:

1. Статьи, набранные в пакете ЛАТ_EX, предоставляются к загрузке через WEB-форму http://intsysjournal.org/generator_form .

2. К статье прилагаются файлы, содержащие название статьи на русском и английском языках, аннотацию на русском и английском языках (не более 50 слов), список ключевых слов на русском и английском языках (не более 20 слов), информация об авторах: Ф.И.О. полностью, место работы, должность, ученая степень и/или звание (если имеется), контактные телефоны (с кодом города и страны), e-mail, почтовый адрес с индексом города (домашний или служебный).

3. Список литературы оформляется в едином формате, установленном системой Российского индекса научного цитирования.

4. За публикацию статей в журнале «Интеллектуальные системы. Теория и приложения» с авторов (в том числе аспирантов высших учебных заведений) статей, рекомендованных к публикации, плата не взимается. Оттиски статей авторам не предоставляются. Журнал распространяется по подписке, экземпляры журнала рассылаются подписчикам наложенным платежом. Условия подписки публикуются в каталоге НТИ «Роспечать», индекс журнала 64559.

5. Доступ к электронной версии последнего вышедшего номера осуществляется через НЭБ «Российский индекс научного цитирования». Номера, вышедшие ранее, размещаются на сайте <http://intsysjournal.org>, и доступ к ним бесплатный. Там же будут размещены аннотации всех публикуемых статей.

Подписано в печать: 19.12.23

Дата выхода: 26.12.23

Тираж: 200 экз.

Цена свободная

Свидетельство о регистрации СМИ: ПИ № ФС77-58444 от 25 июня 2014 г.,
выдано Федеральной службой по надзору в сфере связи, информационных
технологий и массовых коммуникаций (Роскомнадзор).