ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 32 Выпуск 3

ISSN Online 2220-6426
Volume 32 Issue 3

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2020

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

Google Scholar, ULRICHSWEB Global Serials Directory, Russian Science Citation Index, OCLC WorldCat, BASE, OpenDOAR, CYBERLENINKA, DOAJ Directory of Open Access Journals, eLIBRARY.RU, ROAR Registry of Open Access Repositories, Crossref, Math-Net.Ru

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: proceedings@ispras.ru
Сайт: https://ispranproceedings.elpub.ru/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: proceedings@ispras.ru
Web: https://ispranproceedings.elpub.ru/

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2020-32(3)-1

Архитектура системы дедуктивной верификации машинного кода

[4] *И.В. Гладышев, ORCID: 0000-0002-9922-4076 <ilya.v.gladyshev@gmail.com>*
[1,2,3,4] *А.С. Камкин, ORCID: 0000-0001-6374-8575 <kamkin@ispras.ru>*
[1] *А.М. Коцыняк, ORCID: 0000-0003-3499-4368 <kotsynyak@ispras.ru>*
[1,4] *П.А. Путро, ORCID: 0000-0001-9540-8321 <pavel.putro@ispras.ru>*
[1,2,3,4] *А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[3] *Московский физико-технический институт 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9*
[4] *Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В последние годы ИСП РАН разрабатывает систему дедуктивной верификации машинного (бинарного) кода. Мотивация понятна: современные компиляторы, такие как GCC и Clang/LLVM, не застрахованы от ошибок; тем самым, проверка корректности сгенерированного кода (хотя бы для компонентов с повышенными требованиями к надежности и безопасности) не является лишней. Ключевая особенность предлагаемого подхода состоит в возможности переиспользования формальных спецификаций (пред- и постусловий, инвариантов циклов, лемм и т.п.) уровня исходного кода для верификации машинного кода. Инструмент основан на формальной спецификации системы команд и обеспечивает высокий уровень автоматизации: он дизассемблирует машинный код, извлекая его семантику, адаптирует высокоуровневые спецификации для машинного кода и генерирует условия верификации. Система использует ряд сторонних компонентов, включая анализатор исходного кода (Frama-C), анализатор машинного кода (MicroTESK) и SMT-решатель (CVC4). Модульная архитектура позволяет заменять один компонент другим при изменении формата входных данных или используемой техники верификации. В работе рассматривается архитектура инструмента, описывается наша реализация и демонстрируется пример верификации библиотечной функции memset.

Ключевые слова: формальные методы; дедуктивная верификация; анализ бинарного кода; проверка эквивалентности; архитектура системы команд; машинный код; тестирование компиляторов.

Для цитирования: Гладышев И.В., Камкин А.С., Коцыняк А.М., Путро П.А., Хорошилов А.В. Архитектура системы дедуктивной верификации машинного кода. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 7-20. DOI: 10.15514/ISPRAS–2020–32(3)–1

Благодарности. Работа поддержана грантом Минобрнауки РФ RFMEFI60719X0295.

Architecture of a Machine Code Deductive Verification System

[4] *I.V. Gladyshev, ORCID: 0000-0002-9922-4076 <ilya.v.gladyshev@gmail.com>*
[1,2,3,4] *A.S. Kamkin, ORCID: 0000-0001-6374-8575 <kamkin@ispras.ru>*
[1] *A.M. Kotsynyak, ORCID: 0000-0003-3499-4368 <kotsynyak@ispras.ru>*
[1,4] *P.A. Putro, ORCID: 0000-0001-9540-8321 <pavel.putro@ispras.ru>*
[1,2,3,4] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*
[3] *Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia*
[4] *National Research University Higher School of Economics, 20, Myasnitskaya st., Moscow, 101000, Russia*

Abstract. In recent years, ISP RAS has been developing a system for machine (binary) code deductive verification. The motivation is rather clear: modern compilers, such as GCC and Clang/LLVM, are not free of bugs; thereby, it is not superfluous (at least for safety- and security-critical components) to check the correctness of the generated code. The key feature of the suggested approach is the ability to reuse source-code-level formal specifications (pre- and postconditions, loop invariants, lemma functions, etc.) at the machine code level. The tool is highly automated: provided that the target instruction set is formalized, it disassembles the machine code, extracts its semantics, adapts the high-level specifications, and generates the verification conditions. The system utilizes a number of third-party components including a source code analyzer (Frama-C), a machine code analyzer (MicroTESK), and an SMT solver (CVC4). The modular design enables replacing one component with another when switching an input format and/or a verification engine. In this paper, we discuss the tool architecture, describe our implementation, and present a case study on verifying the memset C library function.

Keywords: formal methods; deductive verification; binary code analysis; equivalence checking; instruction set architecture; machine code; compiler testing.

For citation: Gladyshev I.V., Kamkin A.S., Kotsynyak A.M., Putro P.A., Khoroshilov A.V. Architecture of a Machine Code Deductive Verification System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020. pp. 7-20 (in Russian). DOI: 10.15514/ISPRAS–2020–32(3)–1

Acknowledgements. The research was carried out with funding from the Ministry of Science and Higher Education of the Russian Federation (the project unique identifier is RFMEFI60719X0295).

1. Введение

Роль программного обеспечения (ПО) в критической информационной инфраструктуре постоянно растет. В результате сейчас крайне востребованы прикладные методы и инструменты обеспечения корректности наиболее ответственных компонентов ПО. Научным сообществом предложен ряд подходов: некоторые из них ограничиваются проверкой отсутствия в компоненте ошибок определенных типов (например, ошибок времени исполнения), в то время как другие пытаются доказать *полную корректность*, что означает, что все возможные вычисления компонента *завершаются* и удовлетворяют *программному контракту*, выраженному в форме *пред-* и *постусловий* на интерфейсы компонента. Для доказательства такого рода свойств применяют методы *дедуктивной верификации*.

Первые идеи таких методов появились в работах Флойда [1] и Хоара [2] еще в конце 1960-ых годов (индуктивные утверждения, аксиоматическая семантика и прочее). Несмотря на это, верификация промышленного ПО стала *реалистичной* совсем недавно [3-7]. Все известные инструменты дедуктивной верификации императивных программ следуют единому подходу [8]:

- *формально* определяется *семантика* всех операторов языка программирования;
- функциональные требования к компоненту формализуются в виде *пред-* и *постусловий* функций (или методов) на некотором *языке спецификации*;
- пользователем предоставляются дополнительные подсказки для инструмента, такие как *инварианты циклов*, *вспомогательный код* (*ghost code*) и *леммы*;
- на основе спецификаций и подсказок инструментом генерируются *условия верификации*, которые проверяются с помощью *решателя* (*solver*) или *интерактивной системы доказательства теорем* (*proof assistant*);
- доказательство всех условий верификации означает, что *все возможные вычисления* компонента удовлетворяют функциональным требованиям с учетом *предположений* (*гипотез*) о среде исполнения, средствах разработки и т.п.

Одно из предположений состоит в том, что машинный (бинарный) код, сгенерированный компилятором, соответствует семантике языка программирования. Очевидно, что эту гипотезу можно принять только для верифицированных компиляторов, например, CompCert [9], используемых в основном в исследовательских проектах. В индустрии же по-прежнему используют «тяжеловесные» оптимизирующие компиляторы, такие как GCC и Clang/LLVM. К сожалению, они слишком сложны для верификации, и ошибки в сгенерированном машинном коде не являются редкостью [10].

В качестве альтернативного подхода, не принимающего на веру корректность компилятора, мы предлагаем для каждой программы доказывать, что сгенерированный бинарный код удовлетворяет функциональным спецификациям, заданным для исходного кода. Идея выглядит привлекательной – гораздо проще проверить одно конкретное преобразование кода, чем компилятор целиком. Более того, это позволяет использовать агрессивные оптимизации, которые в целом небезопасны, но приемлемы для конкретного компонента и его контракта. В то же время есть много трудностей, которые нужно преодолеть:

- целевая *архитектура* (*система команд*) – регистры, память, режимы адресации и команды – должна быть *формализована* (иначе невозможно математически строго рассуждать о семантике машинного кода);
- *высокоуровневые спецификации* должны некоторым образом *адаптироваться* к бинарному коду (в частности, должно определяться соответствие между переменными в исходном коде и элементами памяти в машинном коде);
- подсказки для инструмента верификации, включая инварианты цикла, вспомогательный код и леммы, должны *переиспользоваться* на уровне бинарного кода (или должен существовать альтернативный способ их задания);
- инструмент должен проверять функциональные свойства получающегося бинарного кода при наличии произвольных *оптимизаций компилятора*.

Оставшаяся часть статьи организована следующим образом. В разд. 2 делается обзор работ, посвященных дедуктивной верификации программ на уровне бинарного кода. В разд. 3 описывается архитектура системы дедуктивной верификации машинного кода. В разд. 4 приведен пример использования предложенного подхода – библиотечная функция `memset`, компилируемая в систему команд RISC-V. Наконец, в разд. 5 делается заключение и обрисовываются направления дальнейших исследований.

2. Обзор литературы

В проекте Why3-AVR [11] с помощью платформы Why3 [12] верифицируются программы без ветвлений и циклов, разработанные на языке ассемблера микроконтроллера AVR. Система команд AVR представляется формально на языке WhyML – синтаксис позволяет описывать команды таким образом, чтобы ассемблерный код (после простого препроцессирования) был допустимым WhyML-текстом. Программист может аннотировать

код пред- и постусловиями и проверять его корректность с помощью внешних решателей или интерактивных систем доказательства теорем. Инструмент может быть полезен для низкоуровневой разработки, поскольку Why3 обладает богатыми возможностями для анализа и преобразования кода. Наш подход отличается: (1) он позволяет повторно использовать спецификации уровня исходного кода для верификации бинарного кода; (2) он масштабируется на более сложные архитектуры за счет использования специализированных языков, таких как nML [13], для спецификации систем команд; (3) он поддерживает циклы в программах (и, соответственно, инварианты циклов в спецификациях).

В работе [14] для верификации машинного кода на подмножествах ARM, PowerPC и x86 (IA-32) используется интерактивная система доказательства теорем HOL4 [15]. Упомянутые архитектуры были формализованы независимо: модели ARM и x86 [16,17] разработаны на HOL4, а модель PowerPC [18] – на Coq [19] (в рамках проекта CompCert [9]), а затем вручную переведена на HOL4. Автор выделяет четыре уровня абстракции: машинный код (уровень 1) автоматически декомпилируется в низкоуровневую реализацию (уровень 2); пользователь вручную разрабатывает высокоуровневую реализацию (уровень 3), а также высокоуровневую спецификацию (уровень 4); доказательство соответствия между этими уровнями гарантирует, что машинный код удовлетворяет высокоуровневой спецификации. Преимущество решения состоит в возможности переиспользования некоторых доказательств (уровни 3 и 4 не зависят от архитектуры). Еще один момент, который следует отметить, – автоматическая трансляция циклов в рекурсивные функции. На наш взгляд, уровень автоматизации можно повысить за счет использования специализированных языков описания систем команд.

Интересный подход к проверке соответствия машинного кода ACSL-спецификациям [20], рассмотрен в работе [21]. Процесс выглядит следующим образом: (1) ACSL-аннотации записываются в виде ассемблерных вставок; (2) модифицированный исходный код транслируется в язык ассемблера; (3) полученный ассемблерный код преобразуется в WhyML; (4) платформа Why3 генерирует условия верификации и проверяет их с помощью внешнего решателя. Метод похож на предлагаемый нами, однако есть существенные отличия. Основное из них заключается в том, что процесс верификации «завязан» на компилятор – при переходе с одного компилятора на другой может потребоваться доработка инструмента. Кроме того, верификация на уровне языка ассемблера не позволяет полностью отказаться от гипотезы о корректности компилятора – ассемблерный код является промежуточным представлением, подлежащим дальнейшей трансляции. Наша цель – сделать инструмент верификации как можно более независимым от компилятора и целевой машины.

В работе [22] демонстрируется возможность переиспользования доказательств корректности исходного кода для верификации машинного кода. Подход иллюстрируется на примере Java-подобного языка, компилируемого в байт-код абстрактной стековой машины. Статья посвящена кодированию с переносом доказательств (PCC – Proof-Carrying Code) и показывает, что (при определенных предположениях) компиляция сохраняет доказательства; другими словами, доказательство корректности исходного кода (построенное автоматически или интерактивно) может быть преобразовано в доказательство корректности машинного кода. Хотя идеи этого подхода могут быть полезны, решаемая нами задача отличается от описанной. Кроме того, предложенное решение привязано к конкретной аппаратной платформе.

3. Предлагаемая архитектура

В этом разделе описывается предлагаемая архитектура системы дедуктивной верификации машинного кода. Система предназначена для проверки машинного кода функции на соответствие спецификациям уровня исходного кода. Инструмент принимает на вход следующие данные:

- верифицированный *исходный код* функции и ее *спецификации* (пред- и постусловия, инварианты циклов и т.п.);
- *неоптимизированный объектный код* функции;
- *оптимизированный объектный код* функции (код, подлежащий проверке);
- *спецификация целевой архитектуры* (формальное описание регистров, режимов адресации и команд микропроцессора);
- *конфигурация компилятора и целевой машины* (размеры типов данных и двоичный интерфейс приложений).

Выходом инструмента является отчет о верификации, содержащий общий вердикт – является ли машинный код функции корректным, – а также частные вердикты для полученных в процессе анализа условий верификации. На рис. 1 изображены компоненты системы и связи между ними.

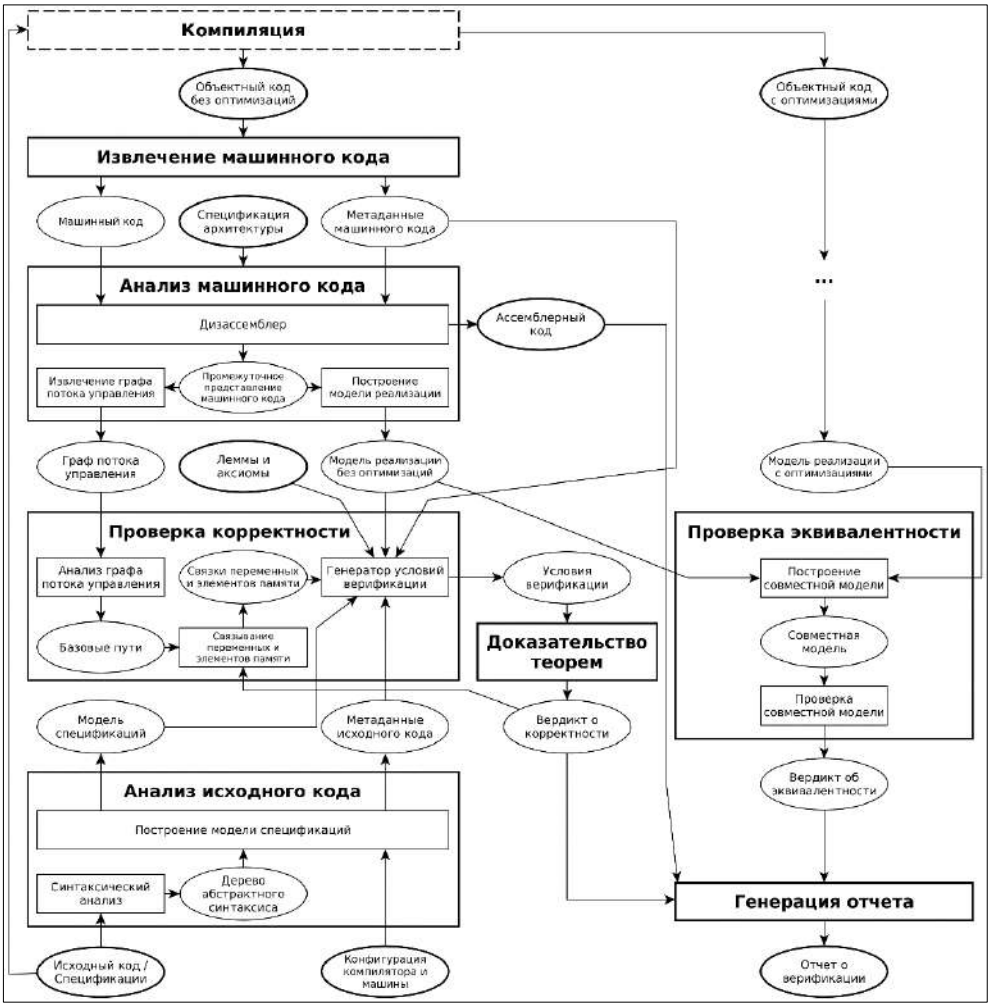


Рис. 1. Предлагаемая архитектура системы дедуктивной верификации машинного кода
Fig. 1. The proposed architecture of the system for deductive verification of machine code

3.1 Модуль извлечения машинного кода

Модуль извлечения машинного кода принимает на вход *объектный файл*, полученный при компиляции *исходного кода*, и выделяет *машинный код* заданной функции с учетом порядка байтов целевой машины. Реализация модуля зависит от формата объектного файла и может использовать существующие утилиты, например, GNU Binutils [23]. Помимо прочего, модуль извлекает из объектного файла вспомогательную информацию (*метаданные*), в том числе таблицу адресов вызываемых функций.

3.2 Модуль анализа машинного кода

Анализ машинного кода в том виде, в каком он есть, ограничивает область применения инструмента одной архитектурой. Более гибкое решение – использование архитектурно-независимого *промежуточного представления*. Для трансляции машинного кода в промежуточное представление используется *дизассемблер*. Это может быть как отдельный инструмент, разработанный для конкретной системы команд, так и модуль, автоматически построенный на основе *спецификации целевой архитектуры*. В спецификации описываются регистры микропроцессора, режимы адресации и команды. Помимо специализированного промежуточного представления дизассемблер выдает *ассемблерный код*, используемый для отладки и формирования отчета о верификации.

На основе полученного промежуточного представления строится *граф потока управления*. Для этого в последовательности машинных команд идентифицируются команды ветвления и вычисляются адреса переходов. За решение этой задачи отвечает *модуль извлечения графа потока управления*. Полученный граф снабжается дополнительной информацией, взятой из спецификации архитектуры, в том числе условиями переходов.

Модуль построения модели реализации переводит промежуточное представление машинного кода в логическую форму, понятную инструментам доказательства теорем. Сложность процедуры зависит от языка представления: если он формализован [16]-[18], в качестве модели может выступать само представление; в противном случае [13] нужны дополнительные преобразования. Для записи модели можно использовать языки, применяемые в системах доказательства теорем, такие как SMT-LIB [24], HOL [15] и Coq [19], или языки, допускающие трансляцию в указанные выше, например, WhyML [12].

3.3 Модуль анализа исходного кода

Синтаксический анализатор получает на вход исходный код функции вместе с ее спецификациями и строит *дерево абстрактного синтаксиса* (*AST – Abstract Syntax Tree*). Дерево отображается в *модель спецификаций* – множество утверждений, задающих функциональные требования (для представления утверждений также можно использовать языки, применяемые в системах доказательства теорем).

Платформы статического анализа позволяют разрабатывать плагины для трансляции исходного кода в требуемые представления. *Модуль построения модели спецификаций* может быть реализован как такой плагин. Для распространенных языков моделирования существуют готовые трансляторы, однако они не всегда подходят для анализа машинного кода (в частности, в них не учитывается двоичный интерфейс приложений).

3.4 Модуль проверки корректности

Основное различие в проверке корректности исходного и машинного кода проявляется на стадии генерации условий верификации. Для исходного кода условия верификации генерируются и доказываются независимо друг от друга. Во время компиляции информация о переменных функции теряется, что делает невозможным использование высокоуровневых

инвариантов циклов в рамках классической схемы генерации условий верификации. В общем случае требуется проверить все возможные соответствия между переменными исходного кода и элементами памяти машинного кода (включая регистры). Если k – число переменных, используемых в циклах, а n – число элементов памяти, задействованных в машинном коде, требуется рассмотреть $k!\,C_n^k$ вариантов.

Для поиска соответствия между переменными и элементами памяти используется *модуль связывания*. Сначала *анализатор графа потока управления* выделяет *базовые пути* – цепочки базовых блоков (в общем случае, ациклические подграфы), покрывающие граф и соединяющие вершины следующих типов: начало функции, конец функции, вход в цикл, выход из цикла. Связывание переменных и элементов памяти осуществляется итеративно, путем поэтапного пополнения множества *связок*: при добавлении связки проверяется истинность условий верификации всех базовых путей, зависимых от связываемой переменной; порядок перебора связок управляется эвристиками.

Ядром системы является *генератор условий верификации*, который для заданного базового пути и заданного соответствия между переменными и элементами памяти строит *условие верификации*, подлежащее доказательству. Часто для успешного доказательства условий верификации требуются вспомогательные *леммы и аксиомы*.

3.5 Модуль проверки эквивалентности

Оптимизированный код, подлежащий верификации, сопровождается *неоптимизированной версией*, полученной из того же исходного кода. При отсутствии оптимизаций сохраняется структура потока управления и, как следствие, инварианты циклов (без этого описанный выше подход не применим). Таким образом, функциональные требования проверяются для кода без оптимизаций, после чего доказывается *эквивалентность* оптимизированной и неоптимизированной версий. Используемый подход к проверке эквивалентности, как и многие другие [25,26], основан на семантическом сопоставлении моделей программ и построении совместной модели (графа совместного исполнения).

3.6 Модуль доказательства теорем

Для доказательства условий верификации используется *модуль доказательства теорем*. В качестве этого модуля можно использовать существующие решатели, как автоматические, так и интерактивные. Основное требование – поддержка битовых векторов и массивов. Данное требование возникает из-за способа моделирования микропроцессоров: (1) регистры и ячейки памяти – битовые векторы; (2) регистровые файлы и блоки памяти – массивы битовых векторов; (3) команды – операции над битовыми векторами.

Табл. 1. Реализация системы дедуктивной верификации машинного кода
Table 1. Implementation of a deductive verification system for machine code

Модуль/формат данных	Реализация	Комментарий
Объектный код	ELF [29]	Популярный формат для представления объектного кода
Модуль извлечения машинного кода	Основан на Binutils [23]	Использует readelf для извлечения машинного кода

Метаданные машинного кода	Таблица адресов функций	Содержит относительные адреса всех функций в объектном файле
Модуль анализа машинного кода	MicroTESK [30], [31]	Дизассемблирование, построение графа потока управления, построение модели реализации
Язык спецификации архитектуры	nML [13]	Описание синтаксиса языка ассемблера, двоичной кодировки и семантики команд микропроцессора
Промежуточное представление машинного кода	MIR	Внутреннее представление MicroTESK
Ассемблерный код	Язык ассемблера	Формат описывается в спецификации архитектуры
Граф поток управления	JSON	Описывает границы базовых блоков, переходы между базовыми блоками и условия переходов
Модель реализации	SMT-LIB 2.6 [24]	Код базовых блоков в SSA-форме
Исходный код и спецификации	C / ACSL [20]	Код на языке Си, аннотированный пред- и постусловиями и инвариантами циклов
Конфигурация целевой машины и компилятора	JSON	Размеры типов данных языка Си и описание двоичного интерфейса приложений
Модуль анализа исходного кода	Frama-C [32] / Why3 [12]	Frama-C – разбор Си и ACSL, плагин – трансляция ACSL в WhyML, Why3 – трансляция ACSL в SMT-LIB
Модель спецификаций	SMT-LIB 2.6 [24]	Пред- и постусловия, инварианты циклов и т.п.
Метаданные исходного кода	JSON	Описывают сигнатуры сгенерированных SMT-LIB функций

Леммы и аксиомы	SMT-LIB 2.6 [24]	Вспомогательные определения для SMT-решателей
Модуль проверки корректности	MicroVer [33]	Основная часть инструмента (см. разд. 3)
Условия верификации	SMT-LIB 2.6 [24]	Инициализация и сохранение инвариантов циклов, выполнимость постусловия
Модуль доказательства теорем	CVC4	Открытый SMT-решатель с поддержкой битовых векторов и массивов
Вердикт о корректности	Обычный текст	“Да”, “нет” или “неизвестно”: если “нет”, то предоставляется контрпример
Модуль проверки эквивалентности	MicroTESK [31]	Проверяет эквивалентность оптимизированного и неоптимизированного кода
Вердикт об эквивалентности	Обычный текст	См. вердикт о корректности
Отчет о верификации	Обычный текст	Результат верификации: результаты проверки всех условий верификации и контрпримеры

4. Апробация подхода

В этом разделе описывается реализация системы дедуктивной верификации машинного кода и ее применение для библиотечной функции memset [6], компилируемой в систему команд RISC-V [27]. В табл. 1 приведены данные о компонентах и форматах ввода-вывода, используемых в реализованной системе.

В табл. 2 представлена информация о функции `memset`: исходный код на языке Си (вместе с ACSL-спецификациями), ассемблерный код и бинарный код. Результаты верификации функции, включая сгенерированные условия верификации и инструменты, необходимые для воспроизведения шагов метода, находятся в открытом доступе [28].

Табл. 2. Исходный, ассемблерный и бинарный код функции `memset`
Tab. 2. Source, assembly and binary code of the `memset` *function*

Исходный код	**Ассемблерный код**	**Бинарный код**
/*@	addi sp, sp, -64	1301 01FC

requires \typeof(s) <: \type(char *);	sd s0, 56(sp)	233C 8102
requires \valid((char *)s+(0..count-1));	addi s0, sp, 64	1304 0104
assigns ((char *)s)[0..count-1];	sd a0, -40(s0)	233C A4FC
ensures \forall char *p;	addi a5, a1, 0	9387 0500
(char *)s <= p < (char *)s + count ==> *p == (char AENO) c;	sd a2, -56(s0)	2334 C4FC
ensures \result == s;	sw a5, -44(s0)	232A F4FC
*/	ld a5, -40(s0)	8337 84FD
void *memset(void *s, int c, size_t count) {	sd a5, -24(s0)	2334 F4FE
char *xs = s;	ld a5, -56(s0)	8337 84FC
/*@	sd a5, -32(s0)	2330 F4FE
loop invariant \valid((char *)xs+(0..count-1));	jal zero, 0xe	6F00 C001
loop invariant \valid((char *)s+(0..\at(count, Pre)-1));	ld a5, -24(s0)	8337 84FE
loop invariant 0 <= count <= \at(count, Pre);	addi a4, a5, 1	1387 1700
loop invariant (char *)s <= xs <= (char *)s + \at(count, Pre);	sd a4, -24(s0)	2334 E4FE
loop invariant xs - s == \at(count, Pre) - count;	lw a4, -44(s0)	0327 44FD
loop invariant \forall char *p;	andi a4, a4, 255	1377 F70F
(char *)s <= p < xs ==> *p == (char AENO) c;	sb a4, 0(a5)	2380 E700
loop assigns count, ((char *)s)[0..\at(count, Pre)-1];	ld a5, -56(s0)	8337 84FC
loop variant count;	addi a4, a5, -1	1387 F7FF
*/	sd a4, -56(s0)	2334 E4FC
while (count-- AENOC)	bne a5, zero, -18	E39E 07FC
*xs++ = (char) AENOC c;	ld a5, -40(s0)	8337 84FD
	addi a0, a5, 0	1385 0700
return s;	ld s0, 56(sp)	0334 8103
}	addi sp, sp, 64	1301 0104
	jalr zero, ra, 0	6780 0000

5. Заключение

Индустрия ПО нуждается в прикладных средствах формальной верификации. В большинстве инструментов анализируется исходный код программ; между тем, поскольку компиляторы могут содержать ошибки, наиболее критичное ПО следует дополнительно проверять на уровне бинарного кода. В этой работе предложена архитектура системы дедуктивной верификации машинного кода и описана конкретная система (реализующая эту архитектуру), позволяющая автоматически проверять машинный код на соответствие ACSL-

спецификациям, заданным для исходного кода на языке Си. Предложенный подход практически независим от целевой архитектуры, поскольку основан на спецификациях системы команд.

Работа по созданию системы не завершена, и многие ее элементы подлежат улучшению. В будущем мы планируем расширить поддержку языка ACSL и пополнить список доступных целевых архитектур (на данный момент специфицированы RISC-V, ARM, MIPS и, частично, Power). Кроме того, мы работаем над практически применимыми методами проверки эквивалентности машинных программ, полученных путем компиляции одного исходного кода с разными параметрами оптимизации. Отдельный вопрос – тестирование системы и разработка репрезентативного тестового набора (к настоящему времени система была испытана на 20 небольших функциях, что явно недостаточно для оценки ее возможностей и ограничений).

Список литературы / References

[1] R.W. Floyd. Assigning Meanings to Programs. Mathematical Aspects of Computer Science. Proceedings of Symposia in Applied Mathematics, vol. 19, 1967, pp. 19-32.

[2] C.A.R. Hoare. An Axiomatic Basis for Computer Programming. Communications of the ACM, vol. 12, issue 10, 1969, pp. 576-585.

[3] G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, G. Heiser. Comprehensive Formal Verification of an OS Microkernel. ACM Transactions on Computer Systems (TOCS), vol. 32, issue 1, 2014, pp. 2:1-2:70.

[4] E. Cohen, W. Paul, S. Schmaltz. Theory of Multi Core Hypervisor Verification. Lecture Notes in Computer Science, vol. 7741, 2013, pp. 1-27.

[5] P. Philippaerts, J.T. Mühlberg, W. Penninckx, J. Smans, B. Jacobs, F. Piessens. Software Verification with VeriFast: Industrial Case Studies. Science of Computer Programming, vol. 82, 2014, pp. 77-97.

[6] D. Efremov, M. Mandrykin, A. Khoroshilov. Deductive Verification of Unmodified Linux Kernel Library Functions. Lecture Notes in Computer Science, vol. 11245, 2018, pp. 216-234.

[7] D.R. Cok. OpenJML: JML for Java 7 by Extending OpenJDK. Lecture Notes in Computer Science, vol. 6617, 2011, pp. 472-479.

[8] A. Kamkin, A. Khoroshilov, A. Kotsynyak, P. Putro. Deductive Binary Code Verification Against Source-Code-Level Specifications. Lecture Notes in Computer Science, vol. 12165, 2020, pp. 43-58.

[9] CompCert Project. Available at: http://compcert.inria.fr, accessed 12.07.2020.

[10] C. Sun, V. Le, Q. Zhang, Z. Su. Toward Understanding Compiler Bugs in GCC and LLVM. In Proc. of the International Symposium on Software Testing and Analysis (ISSTA), 2016, pp. 294-305.

[11] M. Schoolderman. Verifying Branch-Free Assembly Code in Why3. Lecture Notes in Computer Science, vol. 10712, 2017, pp. 66-83.

[12] J.-C. Filliˆatre, A. Paskevich. Why3 – Where Programs Meet Provers. Lecture Notes in Computer Science, vol. 7792, 2013, pp. 125-128.

[13] M. Freericks. The nML Machine Description Formalism. Technical Report TR SM-IMP/DIST/08, TU Berlin CS Department, 1993, 47 p.

[14] M.O. Myreen. Formal Verification of Machine-Code Programs. Ph.D. Thesis. University of Cambridge, 2009, 131 p.

[15] K. Slind, M. Norrish. A Brief Overview of HOL4. Lecture Notes in Computer Science, vol. 5170, 2008, pp. 28-32. DOI: 10.1007/978-3-540-71067-7 6.

[16] A. Fox. Formal Specification and Verification of ARM6. Lecture Notes in Computer Science, vol. 2758, 2003, pp. 25-40.

[17] K. Crary, S. Sarkar. Foundational Certified Code in a Metalogical Framework. Technical Report CMU-CS-03-108. Carnegie Mellon University, 2003, 19 p.

[18] X. Leroy. Formal Certification of a Compiler Back-End or: Programming a Compiler with a Proof Assistant. In Proc. of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of programming languages (POPL), 2006, pp. 42-54, DOI: 10.1145/1111037.1111042.

[19] Y. Bertot. A Short Presentation of Coq. Lecture Notes in Computer Science, vol. 5170, 2008, pp. 12-16.

[20] P. Baudin, P. Cuoq, J.-C. Filliˆatre, C. March´e, B. Monate, Y. Moy, V. Prevosto. ACSL: ANSI/ISO C Specification Language. Version 1.13, 2018, 114 p.

[21] T.M.T. Nguyen, C. March´e. Hardware-Dependent Proofs of Numerical Programs. Lecture Notes in Computer Science, vol. 7086, pp. 314-329.
[22] G. Barthe, T. Rezk, A. Saabas. Proof Obligations Preserving Compilation. Lecture Notes in Computer Science, vol. 3866, pp. 112-126.
[23] GNU Binutils. Available at: https://www.gnu.org/software/binutils, accessed 12.07.2020
[24] C. Barrett, P. Fontaine, C. Tinelli. The SMT-LIB Standard Version 2.6. Release 2017-07-18. 104 p.
[25] M. Dahiya, S. Bansal. Black-Box Equivalence Checking Across Compiler Optimizations. Lecture Notes in Computer Science, vol. 10695, pp. 127-147.
[26] B. Churchill, O. Padon, R. Sharma, A. Aiken. Semantic Program Alignment for Equivalence Checking. In Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2019, pp. 1027-1040.
[27] RISC-V Foundation. Available at: https://riscv.org, accessed 12.07.2020.
[28] Results of memset binary code verification. Available at: https://forge.ispras.ru/attachments/7472, accessed 12.07.2020.
[29] Executable and Linkable Format (ELF. Available at: http://www.skyfree.org/linux/references/ELF Format.pdf, accessed 12.07.2020.
[30] M. Chupilko, A. Kamkin, A. Kotsynyak, A. Protsenko, S. Smolov, A. Tatarnikov. Test Program Generator MicroTESK for RISC-V. In Proc. of the International Workshop on Microprocessor and SOC Test and Verification (MTV), 2018, 6 p.
[31] MicroTESK Framework. Available at: http://www.microtesk.org, accessed 12.07.2020.
[32] Frama-C Platform. Available at: http://frama-c.com, accessed 12.07.2020.
[33] MicroVer Project. Available at: https://forge.ispras.ru/projects/microver, accessed 12.07.2020.
[34] CVC4 Solver. Available at: https://github.com/CVC4/CVC4, accessed 12.07.2020.

Информация об авторах / Information about authors

Илья Владимирович ГЛАДЫШЕВ – студент магистратуры по направлению «Системное программирование» в ВШЭ. Сфера научных интересов: формальная верификация и операционные системы.

Ilya Vladimirovich GLADYSHEV is a Master's student studying system programming at the National Research University "Higher School of Economics". His research interests include formal verification and operating systems.

Александр Сергеевич КАМКИН – кандидат физико-математических наук, ведущий научный сотрудник отдела технологий программирования ИСП РАН; преподает в МГУ, МФТИ и ВШЭ. Область научных интересов: проектирование цифровой аппаратуры, верификация и тестирование, статический и динамический анализ HDL-описаний, высокоуровневый синтез.

Alexander Sergeevich KAMKIN is a leading researcher at the Software Engineering Department of Ivannikov Institute for System Programming of the Russian Academy of Sciences (ISP RAS). He is also a lecturer at Moscow State University (MSU), Moscow Institute of Physics and Technology (MIPT), and Higher School of Economics (HSE). His research interests include digital hardware design, verification and testing, static and dynamic analysis of HDL descriptions, and high-level synthesis. Alexander has a PhD in Physics and Mathematics.

Артем Михайлович КОЦЫНЯК младший научный сотрудник отдела технологий программирования. Область научных интересов: верификация и тестирование, компиляторные технологии, высокоуровневый синтез и формальные методы.

Artem Mikhailovich KOTSYNYAK is a junior researcher at the Software Engineering Department of Ivannikov Institute for System Programming of the Russian Academy of Sciences (ISP RAS). His research interests include verification and testing, compiler technologies, high-level synthesis and formal methods.

Павел Андреевич ПУТРО получил степень бакалавра в области программной инженерии и степень магистра в области системного программирования в ВШЭ, Москва, Россия. Работает ИСП РАН. Исследовательские интересы включают дедуктивную верификацию, логическое программирование и статический анализ машинного кода.

Pavel Andreevich PUTRO received a bachelor's degree in software engineering and a master's degree in system programming from the National Research University Higher School of Economics, Moscow, Russia. He works at the Ivannikov Institute for System Programming of the RAS. His research interests include deductive verification, logic programming, and machine code static analysis.

Алексей Владимирович ХОРОШИЛОВ, ведущий научный сотрудник, кандидат физико-математических наук, директор Центра верификации ОС Linux в ИСПРАН, доцент кафедр системного программирования МГУ, ВШЭ и МФТИ. Основные научные интересы: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей, методы анализа требований, операционная система Linux.

Alexey Vladimirovich KHOROSHILOV, Leading Researcher, Ph.D. in Physics and Mathematics, Director of the Linux OS Verification Center at ISP RAS, Associate Professor of System Programming Departments at Moscow State University, the Higher School of Economics, and Moscow Institute of Physics and Technology. Main research interests: design and development methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing, requirements analysis methods, Linux operating system.

DOI: 10.15514/ISPRAS-2020-32(3)-2

Modeling of library functions in an industrial static code analyzer

[1] *M.V. Belyaev, ORCID: 0000-0003-3489-3508 <mbelyaev@ispras.ru>*
[2]*E.S. Romanenkov, ORCID: 0000-0001-9736-5492 <esromanenkov@ispras.ru>*
[1,2]*V.N. Ignatyev, ORCID: 0000-0003-3192-1390 <valery.ignatyev@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*

Abstract. SharpChecker is an industrial level static analyzer, which is aimed at detection of various bugs in C# source code. Because the tool is actively developed, it requires more and more precise information about program environment, especially about results and side-effects of library functions. The paper is devoted to the evolution of models for the standard library historically used by SharpChecker, its advantages and drawbacks. We have started from SQLite database with the most important functions properties, then introduced manually written C# model implementations of frequently used methods to add support of data container states and have recently developed a model, built by a preliminary analysis of library source code, which allows to gather all significant side-effects with conditions for almost whole C# library.

Keywords: static analysis; library; source code analysis

For citation: Belyaev M.V.**,** Romanenkov E.S.**,** Ignatyev V.N. Modeling of library functions in an industrial static code analyzer. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 21-32. DOI: 10.15514/ISPRAS–2020–32(3)–2

Моделирование библиотечных функций в промышленном статическом анализаторе кода

[1] *М.В. Беляев, ORCID: 0000-0003-3489-3508 <mbelyaev@ispras.ru>*
[2] *Е.С. Романенков, ORCID: 0000-0001-9736-5492 <esromanenkov@ispras.ru>*
[1,2] *В.Н. Игнатьев, ORCID: 0000-0003-3192-1390 <valery.ignatyev@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*

Abstract. SharpChecker – это статический анализатор промышленного уровня, предназначенный для обнаружения различных ошибок в исходном коде C#. Поскольку инструмент активно разрабатывается, ему требуется все более точная информация о программной среде, особенно о результатах и побочных эффектах функций библиотеки. Статья посвящена эволюции моделей для стандартной библиотеки, исторически используемой SharpChecker, ее преимуществам и недостаткам. Мы начали с базы данных SQLite с наиболее важными свойствами функций, затем добавили написанные вручную реализации модели C# часто используемых методов для поддержки состояний контейнера данных, а недавно разработали модель, построенную на основе предварительного анализа исходного кода библиотеки,

которая позволяет собрать все существенные побочные эффекты с условиями для почти всей библиотеки C#.

Ключевые слова: статический анализ; библиотека; анализ исходного кода

Для цитирования: Беляев М.В., Романенков Е.С., Игнатьев Н.В. Моделирование библиотечных функций в промышленном статическом анализаторе кода. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 21-32 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–2

1. Introduction

There are four main methods for finding defects in programs: manual inspection, testing, dynamic analysis and static analysis. Testing requires to prepare a large number of tests, but even if the code is fully covered by tests, some part of program execution cases is left unconsidered.

Fuzz testing, or fuzzing, is automatic generation of random correct and incorrect input data and running the program on that input data with error detection. Fuzzing requires a large number of program runs, and its applicability is limited by the difficulty of finding the input data that causes an error.

Dynamic analysis also requires to prepare an execution environment and a set of input data, and it covers only the paths reachable on given input data.

Static analysis allows to find defects in programs without execution. The result of a static code analyzer is a list of potential defects found in the program with defect type, origin location and propagation trace.

Source-level static analysis allows the analyzer to get more information about the source code, including variable types and syntax, that is lost after transforming the program into the compiler's intermediate representation. Such information can be used to increase the quality of analysis and to report the defect trace precisely mapped to the source code. Moreover, there are several types of errors that could be discovered only using static analysis, for example, unreachable code, copy-paste errors and comparison of a pointer with \texttt{null} after dereferencing it.

Most projects use external libraries in binary form, so source-level static analyzer is unable to get any information about such functions and the resulting analysis quality is significantly reduced. To be able to produce precise results it is necessary to provide the most sensitive information about called external functions, for example, which exceptions can be triggered with their conditions, which values are changed, etc. This paper is devoted to the possible methods for data provision that were tested with the static analysis tool SharpChecker [1], which is also available as a part of Svace [2], [3] static analyzer tool-set. SharpChecker contains several analysis engines: AST, dataflow, symbolic execution [4], taint [5] and experimental machine learning based subsystem [6].

Modeling of external libraries code is an important task in interprocedural source code analysis. Knowing the preconditions, return values and side effects of external functions can improve the precision and recall of analysis in general. Programming languages such as Java and C# have large standard libraries – Java Class Library and .NET Framework (or .NET Core) respectively, which are widely used by programmers. So this problem gets high priority even for initial analyzer version.

We will demonstrate the evolution of our approaches, that were used by SharpChecker in historical order. At each stage the specified method was sufficient to meet the requirements of the analyzer. During the evolution of algorithms, used by SharpChecker, or because of the development of detectors for new error types it becomes necessary to get higher precision of modeling, to cover huge amounts of new libraries, to model specific properties of several functions. Since SharpChecker is the industrial level tool which is deployed into large companies, we also have to consider such criteria as performance, results stability and determinism. As a result, we can't be limited by the single approach and will show how we combine all of them to achieve better results.

The paper is organized as follows. In Section 2 we define the set of metrics and properties used to evaluate and compare the described techniques. Section 3 is devoted to the existing and well-tested

approaches. In Section 4 we describe the newly developed approach that we plan to finish and integrate into production. We present most serious problems that we have solved and address. Evaluation of the proposed approach using artificial examples and a representative set of opensource projects, containing more than 4 million lines of code (LOC) from 24 projects, is shown in Section 5. In the last Section 6, we summarize the results of the work and present
directions of future research.

2. Comparison criteria

Although SharpChecker uses several approaches for library modeling, it may be due to historical and other reasons, which are not based on the actual drawbacks of any specific technique. We propose the set of criteria that we will use for comparison in the conclusion. We classify the following list as the most important properties.

- *Scalability* – the approximate number of methods that are modeled using the specific approach. We measure only the order of magnitude of the corresponding value.
- *Performance* – the influence of resulting function model on the analyzer performance. Since it is hard to measure it, we will sort all methods and use comparative order as a value.
- *Completeness* is a logical property which clarifies if the given approach covers all method properties or it will be necessary to add something in future. For example, it is possible to specify only a single property of a method, such as whether it can return `null`. It will require a way to specify additional properties, such as precondition for `null` return value, or that method also uses locks or creates a resource that should be managed.
- *Maintainability* means how easy it is to update or correct existing models, when a new library version is published or to supplement with models for newly added functions.
- *Manual correction* – a property which specifies who is able to correct models manually: analyzer developer, user, both or nobody.
- *Size* – the total size in bytes of all models. The average size of a model of a given method could be calculated as a result of division of given value by the total number of methods, covered by the approach.
- *Private code* – a logical property, which means if it is possible for the user to add data about private libraries without sharing it with analyzer developers.

All listed metrics are calculated for SharpChecker but the order of magnitude of resulting values is valid for general case because of the nature of the analyzed approaches.

3. Approaches

3.1 Hardcoded semantics

The simplest approach for modeling individual properties of several methods, which was used in the earliest versions of analyzer, is *hardcoded semantics*. It is a very flexible method, because it makes it possible to model almost everything. It is still used by SharpChecker. For example, the semantics of the following functions is hardcoded in the analysis engine:

`string.IsNullOrEmpty(string);`

`string.IsNullOrWhitespace(string);`

various built-in, NUnit and xUnit asserts;

`System.Environment.Exit(int);`

`System.Nullable<T>.HasValue;`

`System.Nullable<T>.Value` and so on.

The main drawback of this approach is inability to scale for substantial share of methods that is necessary to cover, because writing support for each function in different parts of analysis engine is

rather difficult. At the same time it allows to represent some unique properties, for example, that `System.Environment.Exit(int)` terminates the execution and that `System.Nullable<T>.HasValue` doesn't dereference `this` and instead compares it with `null`. Since the number of such properties and corresponding methods is very small and these properties are very different, so that the unified approach is not possible, and hardcoding is the only way to cover such situations.

3.2 Property database

One of the earliest and most important detectors of SharpChecker is related to null dereference. It requires to know, for example, if some method throws `ArgumentNullException`, when it's argument is `null`. For the majority of functions such information is provided in documentation. As a result, the first approach to library function modeling implemented in SharpChecker [1] is extracting some properties of methods from the documentation at MSDN, which is now replaced with Microsoft Docs [7], and storing it in per-method XML files. The extracted properties include:

- method signature, which includes its fully qualified name and types of parameters;
- a list of possible thrown exception types;
- the possibility of returning null;
- which parameters must be non-null.

Other properties are manually added for some methods. Such method properties specify whether a described method:

- disposes `this`;
- enters or exits a synchronization monitor;
- changes the inner state of `this`;
- is *pure* (has no side effects and returns the same value if called with the same arguments multiple times);
- uses multithreading;
- is an obsolete cryptographic algorithm implementation;
- is a virtual method which must be called in any method that overrides it.

Later thousands of XML files were transformed into an SQLite database of method properties, because such database allows more compact storage and faster lookup. Boolean fields are represented as bit flags in a single `Attributes` integer column.

This approach allows a rather compact storage of method annotations, doesn't depend on the availability of library source code, covers all methods that have available documentation at the moment of parsing and allows easy manual correction of annotations.

However, it doesn't represent a large part of method semantics, including exception conditions, field assignments, return value conditions, etc., and adding any new information field requires the developers to fill it by hand for all known methods. When new methods are added into .NET Framework, manual fields should also be filled.

An additional significant issue with such database is that it becomes outdated and it's hard to transfer manually added fixes and new information into updated version. Even if we overcome this difficulty, we cannot verify that the resulting database doesn't contain parse errors, which could suddenly appear only on customers' code.

3.3 Code models

C# programs widely use collections – data structures and methods for data manipulation. C# programming language itself supports LINQ [8] – SQL-like language which is used as a unified API for collections library. For path-sensitive analysis it is very important for the analyzer to be able to understand if any specific collection could be empty or not, because a lot of errors are related with

corner cases, when loop iteration over collection will not perform even a single iteration. Listing 1 demonstrates an example of such situation, when `lastSatisfied` could have value `null` if `list` has no elements. However, it can never happen, because an element is always added to the list before the loop, so it's never empty. It is impossible to reuse both described approaches: there are too many types and interfaces to hardcode all of them, and it is impossible to hold internal state of object, which represents a collection, using a database.

```
1  private bool check(object obj) => true;
2  private void foo(IList<object> list, object elem)
3  {
4     object lastSatisfied = null;
5    list.Add(elem);
6    foreach (var elem in list)
7    {
8       if (check (elem))
9          lastSatisfied = elem;
10    }
11    Console.Writeline(lastSatisfied.ToString());
12 }
```

Listing 1. NRE defect example

To overcome the described problem, we have developed another approach that still doesn't require the source code of libraries to be available. This approach is to model the most frequently used methods with handwritten C# code. It isn't necessary for such implementation to be a complete drop-in replacement of the corresponding library method. Instead, such code contains only some key features useful for analysis, such as management of the internal state of an object, exception conditions, data flow between arguments, fields and return value, the possibility of returning null etc. When the analysis engine is improved, it becomes able to analyze such models better, and so the quality of library modeling improves even though the models are not changed.

The implementation of the given method is not very hard. The models are organized similarly to their implementation in the standard library using identical names. All models are joined into the single C# project, which is analyzed before the target code. So, analyzer can differ these implementations from user code only because of special labels and prefer to apply summaries of models instead of using library database.

The main disadvantage of this approach is that the models are written manually, and that means that they cover only a small fraction of .NET Framework methods. Another theoretically possible drawback is that if we managed to write manual models for a large number of methods, SharpChecker would spend a lot of time analyzing these models in the beginning of each run. This disadvantage is important especially for using SharpChecker with small projects. Such modeling is very significant for taint analysis engine [5], because it makes it possible to specify paths of tainted data propagation and cleanup between arguments, object fields and return value.

The described approach seems to have an evident extension. What if it would be possible to perform preliminary analysis of open source standard libraries, save its results similarly to a regular user method and reuse for all further static analyzer runs? To make it possible to explore the proposed approach it is necessary to solve several serious problems. First of all, regular code of library methods implementation is much more complex than used for models, so the analyzer should be very precise, stable and deterministic, because an error of analysis of library will be distributed and multiplied into dozens of errors on target user code. Analyzer must also be very efficient, because the conditions of every interesting property, gathered from real implementation, are big and complex. Analyzer should be able to generate compact summaries, because the summaries of preliminarily analyzed libraries become a part of tool distribution package. It took several years to overcome mentioned problems, and the most interesting issues and its solutions are presented in the next section.

4. Library pre-analysis

Growing collaboration of Microsoft with open-source community has lead to development of .NET Core – a cross-platform open-source runtime and class library for .NET programming languages. .NET Core is partially compatible with .NET Framework, although now it doesn't support all classes, methods and features of .NET Framework. .NET 5, which is planned to be released this year, will join .NET Core and .NET Framework into one open platform.

The SharpChecker's interprocedural analysis engine works by traversing the call graph in the reverse topological order, analyzing methods from callees to callers and saving all the knowledge that it has gained about a method into a method summary, which can be kept in memory or written into a file. When the analyzer encounters a call of an already analyzed method, it *applies* its summary to bring that knowledge into the context of current method.

A fully automatic approach to library modeling is to analyze the source code of a library (.NET Core in this case, but the technique could be used with any library with source code) with SharpChecker and to save all summaries of the library methods into a file. Then, when analyzing a user's project, SharpChecker loads the summaries of called library methods from the file and applies them.

Like the previous approach, the quality of library modeling is closely connected with the quality of the main analysis engine. This property has the following consequences:

- when the analysis engine is improved, the quality of library modeling increases;
- item improvements that are aimed to improve the quality of models make the main analysis better;
- item to get good results when using library summaries, it is necessary to fix many previously unknown bugs in the analysis engine and even to implement some new features.

4.1 Summary size reduction

Before the introduction of pre-analysis for library modeling, SharpChecker used method summaries during analysis to keep gathered information and cleared summary after the analysis of all callers. The first implementations used identical summaries that were serialized and stored on disk. The resulting file has 0.5 gigabyte size for 137 thousand records. Initially the size was even more, because it contained multiple instances of summaries for each method, as some statistical information is changed after the first serialization and needs to be saved again, and it's impossible to remove previously serialized record from a compressed archive without rewriting the whole archive. To use the approach in production it would be necessary to add that huge file into distribution package, and it is inconvenient to use and update.

A method summary consists of several components, gathered by the common analysis engine and separate error detectors. Analysis of the biggest summaries showed that a lot of space is occupied by the information, required for statistics-based detection of possible null return value dereferences. Since it is not necessary to discover errors in the library during the analysis of the user code, all statistical data for library methods were removed. Summaries of all private methods, which are not available from the user code, were also removed. All these fixes allowed to reduce the total size to 230 MB for 80 thousand records.

Currently we are developing additional more complex features which will also decrease the total file size, such as simplification of stored conditions.

4.2 Performance issues

Previously used methods for modeling the majority of library functions were unable to store pre-conditions. Despite of them pre-analysis produces tons of conditions for every property. Joined with user conditions they reach performance limits. For example, consider the example at Listing 2. Modeling based on the pre-analysis will generate condition `arg1 != null`

`\&\& arg2 == null` instead of the simpler form `arg2 == null`, build by the previous library modeling subsystem. Moreover, every condition, obtained as a result of analysis of `foo` will contain `arg1 != null \&\& arg2 != null`, significantly increasing the complexity of every further condition.

```
1. void foo(object arg1, object arg2)
2. {
3.      if (arg1 == null)
4.         throw new ArgumentNullException();
5.      if (arg2 == null)
6.         throw new ArgumentNullException();
7.
8.      …
9. }
```

Listing 2. An example of a method throwing `ArgumentNullException`

To address the problem, we develop condition simplification methods. Since all conditions are automatically generated, they have a lot of redundancy. Manual inspection of generated conditions showed that *variable substitution* will remove a lot of useless conditions. We extract the known values from equality for conjunction or its negation for disjunction and substitute value into other components of condition. But conditions, generated by analyzer are very complex. The current limit for every condition is 200000, what means that a condition tree can have 200000 nodes. Condition tree is a non-binary tree, where leaves represent atomic conditions, such as `x = 5`, and non-leaf nodes contain one of three operations: negation, conjunction or disjunction. Thus simplification should be carefully preformed, because it is necessary to prevent multiple traverse of such huge conditions.

4.3 New features

An example of a feature that has not been implemented yet is field sensitivity. If `string.IsNullOrEmpty(string)` method was implemented as Listing 3 shows the analyzer would correctly `understand' its behavior from summaries, removing the necessity of hardcoded semantics.

```
1. static bool IsNullOrEmpty(string value)
2. {
3.      return value == null ||
4.         value == string.Empty;
5.
```

Listing 3. Simple implementation of `string.IsNullOrEmpty`

However, in .NET Core this method has following implementation [9] shown in Listing 4.

```
1. static bool IsNullOrEmpty(string value)
2. {
3.      return (value == null ||
4.         0u >= (uint)value.Length) ? true : false;
5. }
```

Listing 4. The original implementation of `string.IsNullOrEmpty`

The comments say that such trick with a ternary operator and >= instead of == are used to workaround some performance issues with the current JIT compiler. Analysis of such implementation discovered a bug in the control flow graph, where the logical disjunction operator affected only the control flow, but the value for its result was not created. Even after fixing this issue, the analyzer can't infer that `string.IsNullOrEmpty(x) && x ==`

`"Hello, world!"` is false, because it lacks some kind of field sensitivity: `a = b` ⇏ `a.f = b.f`.

Using summaries allows the analyzer to provide warning traces inside library functions that make it more easy for the user to understand the reason of the defect and to determine if the warning is true positive or false positive. It is similar to Microsoft SourceLink [10] technology that allows to browse and step library source code when debugging.

5. Practical evaluation

The current state of summary pre-analysis implementation doesn't allow to deploy it to the main analyzer branch, since it has significant regressions in analysis results. We consider separate major error detectors consequently to decrease the number of false positives, which were produced by the approach. The results of evaluation are presented for the `NRE.*.ARGUMENT` checker family that should be one of the most beneficial from the new technique.

5.1 Artificial examples

Evaluation on existing set of tests used during SharpChecker development showed that the analyzer, unfortunately, fails more tests with the pre-analysis summaries than without any function models (except hardcoded). One of the causes is that we focused on null dereference checker during the development of library pre-analysis approach, and some other checkers are either accidentally broken or not properly supported. However, even if we filter only null dereference tests, we still get more failures (18 versus 10) when using pre-analysis summaries. Some tests are specifically designed to check warnings for library methods, which may return null, and now they don't pass because the analyzer doesn't distinguish between library and source methods. Another cause is that this approach is currently in early preview stage, and there are many things to improve, because this method is very sensitive to any analysis errors.

5.2 Open source projects

Evaluation on a set of 24 open-source projects confirmed that the analyzer is not yet ready to use summary pre-analysis instead of the database. Among new null dereference warnings there are many false positives, and some true positives disappeared. However, there are some new true positive warnings which were not found before due to outdated and incomplete database, and the trace that shows where inside the library the null value is dereferenced is useful for reviewing expert. Some warning changes seem to be accidentally introduced analyzer imprecisions or bugs, because these warnings are not related to external methods at all. We will continue our efforts in development of summary-based approach to make it production-ready, as it has significant advantages over other approaches, and fixing issues that arise during the analysis of libraries helps to improve the analysis quality for user code.

As for the performance, the total slowdown of summary-based analysis is about 1 hour (≈60%) on this project set in comparison with property database approach. It may be noted that the slowdown for single project may be slightly less, because for a set of projects the summaries are loaded from file and deserialized multiple times. However, most time is spent due to increased complexity of analysis, because the analyzer has significantly more information about methods, such as value flow and exception conditions.

5.3 Related work

ReSharper [11] (a code analysis and refactoring plugin for Visual Studio) and Rider [12] (a .NET IDE from JetBrains, which is based on ReSharper analysis engine) use XML annotation files for .NET Framework, .NET Core and many widespread libraries, such as Xamarin, Entity

Framework, NUnit, xUnit, Newtonsoft.Json, log4net etc. Currently annotations allow to specify such properties of methods [13]: whether a method can return null, is pure, invokes passed delegate, takes a path as a parameter, is implicitly used through reflection, modifies a collection, is an assertion with condition, terminates the control flow. Annotations can also define method contracts, which specify the dependencies between method input (parameters) and output (returned value, out parameters and control flow effect). JetBrains annotations are documented, regularly updated and released under MIT license, so they should be considered for usage in SharpChecker as a free source of quality improvement.

Coverity static analyzer [14] allows users to specify the behavior of external functions using code models, that are similar to our models [15]. Such models are compiled and analyzed like regular code, but they have special builtin function invocations that represent specific features, for example, `nondet()` represents an unknown condition, `unknown()` represents an unknown value and `UseAfterFreePrimitives.use(x)` represents a usage of a value. Coverity can produce such models automatically by analyzing source code with `cov-make-library` command, and a set of models for Java and Android classes is bundled with the analyzer. Although this approach allows to use both automatically generated and manually written models, it has the following disadvantages:

- generating models from the internal representation of analysis data loses some information in comparison with serialized summaries;
- item constant effort of the analyzer's developers is required to write and support code that generates the model for each feature;
- item compilation and analysis of models consumes additional time, though it's less than time required for the analysis of original source code.

Some languages and frameworks have built-in support for annotating code properties and checkers in the compiler that verify these properties. For example, C# adds `nullable reference types` analysis, that makes all reference types non-nullable by default and requires the user to mark all variables which may hold `null` with a `?` postfix. These annotations are preserved in compiled libraries and allow to check, for example, whether a function can return null or can take null as a parameter value. `AllowNull`, `MaybeNullWhen`, `NotNullWhen` and other attributes [16] allow to specify more complex nullability cases. However, conformance to the annotations is not enforced: the compiler emits warnings (not errors) when it detects violations, and these warnings may be suppressed with a postfix `!` operator. So, a static analyzer should ignore nullability annotations for user code and use them for library functions only when there are no other models. Java uses `@Nullable` and `@NotNull` annotations for this purpose, but it adds a third state~-- not annotated. The fact that nullability annotations are integrated into the languages and checked by the compiler stimulates programmers to annotate their code; on the contrary, a developer that doesn't use a particular static analyzer wouldn't provide annotations for his library for this static analyzer. However, these attributes, annotations and language features describe only some properties like nullability, and do not replace proper modeling of library functions.

6. Conclusion

The paper presents all pros and cons of different approaches used by SharpChecker. Major benefits and drawbacks of all described approaches are summarized in Table 1.

Table 1. Benefits and drawbacks of described approaches

	Hardcoded	Database	C# model	Pre-analysis
Scalability	10^1	10^4	10^2	10^4
Performance	0	1	2	3

Completeness	–	–	–	+
Unique prop	+	–-	+	+
Maintainability	-	-	-	+
Manual corr.	Developer	Both	Both	Nobody
Size	N/A	27 MB	0.8 MB	230 MB
Private code	-	-+	-	+

Table 1 demonstrates that pre-analysis approach is more flexible and covers all features of all other approaches. The major drawback of the technique is significant performance degradation. But the resulting models always have quality suitable for analyzer, because models are generated by the same algorithms and the quality will increase together with improvement of analysis engine itself. It covers all libraries, since most of them have opened sources. Library models can be easily updated and moreover created from private sources by the user independently. The additional advantage of the approach is predictability~-- it is unlikely to get very poor results due to parse or human error that could suddenly appear on inaccessible code. All algorithms, used for generation and application are tested twice, similarly to a compiler bootstrap.

The production version of SharpChecker now uses hardcoded semantics and a database together, and used C# models before recent changes. The authors think that usage of four different ways for library modeling is redundant, so when analyzer is ready and the pre-analysis approach is tested enough, it will replace all others completely. But it doesn't mean that this way is the only right technique, because it is impossible to use it for earlier versions, when analyzer was not able to extract and reuse data precisely and efficiently and, moreover, didn't require such high accuracy of modeling.

Further improvement of library modeling may be the analysis of library binaries. Since C# uses CIL [17] as an intermediate representation and it could be decompiled back to C# source code with a reasonable quality and processed as regular sources, so it doesn't require a separate analysis engine for CIL. The most advanced use case can include decompilation and analysis of binary libraries on the fly.

References / Список литературы

[1]. V. K. Koshelev, V. N. Ignatiev, A. I. Borzilov, and A. A. Belevantsev. SharpChecker: Static analysis tool for C# programs. Programming and Computer Software, vol. 43, issue 4, 2017, pp. 268-276.

[2]. V. P. Ivannikov, A. A. Belevantsev, A. E. Borodin, V. N. Ignatiev, D. M. Zhurikhin, and A. I. Avetisyan. Static analyzer Svace for finding defects in a source program code. Programming and Computer Software, vol. 40, issue 5, 2014, pp. 265-275.

[3]. A. Belevantsev, A. Borodin, I. Dudina, V. Ignatiev, A. Izbyshev, S. Polyakov, E. Velesevich, and D. Zhurikhin. Design and development of Svace static analyzers. In Proc. of the 2018 Ivannikov Memorial Workshop (IVMEM), 2018, pp. 3-9.

[4]. Koshelev V., Dudina I., Ignatyev V., Borzilov A. Path-Sensitive Bug Detection Analysis of C# Program Illustrated by Null Pointer Dereference. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp.59-86 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-5 / Кошелев В.К., Дудина И.А., Игнатьев В.Н., Борзилов А.И. Чувствительный к путям поиск дефектов в программах на языке C# на примере разыменования нулевого указателя. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 59-86

[5]. M.V. Belyaev, N.V. Shimchik, V.N. Ignatyev, and A.A. Belevantsev. Comparative analysis of two approaches to static taint analysis. Programming and Computer Software, vol. 44, issue 6, 2018, pp. 459-466.

[6]. G. Morgachev, V. Ignatyev, and A. Belevantsev. Detection of variable misuse using static analysis combined with machine learning. In Proc. of the 2019 Ivannikov ISP RAS Open Conference (ISPRAS), 2019, pp. 16-24.

[7]. .NET Framework API Reference. Available at: https://docs.microsoft.com/en-us/dotnet/api/?view=netframework-4.5. Accessed: Apr. 10, 2020.

[8]. E. Meijer, B. Beckman, and G. Bierman. LINQ: Reconciling object, relations and XML in the .NET Framework. In Proc. of the 2006 ACM SIGMOD International Conference on Management of Data, 2006, p. 706.
[9]. Source code implementation for string.IsNullOrEmpty(). Available at: https://github.com/dotnet/coreclr/blob/1f3f474a13bdde1c5fecdf8cd9ce525dbe5df000/src/System.Private.CoreLib/shared/System/String.cs#L439-L448. Accessed: Apr. 10, 2020.
[10]. Source Link – a language- and source-control system for providing source debugging experiences for binaries. Available at: https://github.com/dotnet/sourcelink/blob/master/README.md. Accessed: Apr. 10, 2020.
[11]. Features – ReSharper. Available at: https://www.jetbrains.com/resharper/features/. Accessed: May 18, 2020.
[12]. Features – Rider. Available at: https://www.jetbrains.com/rider/features/. Accessed: May 18, 2020.
[13]. External Annotations – Help ReSharper. Available at: https://www.jetbrains.com/help/resharper/Code_Analysis__External_Annotations.html. Accessed: May 18, 2020.
[14]. Coverity Static Analysis. Available at: https://www.synopsys.com/content/dam/synopsys/sig-assets/datasheets/SAST-Coverity-datasheet.pdf. Accessed: May 18, 2020.
[15]. Coverity 2018.09 Command Reference. Available at: https://www.academia.edu/38375284/Cov command ref. Accessed: May 18, 2020.
[16]. C# Reserved attributes: Nullable static analysis. Available at: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/attributes/nullable-analysis, Accessed: May 18, 2020.
[17]. CIL – Common Intermediate Language. Available at: https://en.wikipedia.org/wiki/Common Intermediate Language. Accessed: Apr. 10, 2020.

Information about authors / Информация об авторах

Mikchail Vladimirovitch BELYAEV – Intern Researcher. Research interests: compiler technologies, static code analysis.

Михаил Владимирович БЕЛЯЕВ – стажер-исследователь. Научные интересы: компиляторные технологии, статический анализ кода.

Egor Sergeevitch ROMANENKOV – Master student of the CMC faculty. Research interests: compiler technologies, static code analysis.

Егор Сергеевич РОМАНЕНКОВ – студент магистратуры факультета ВМК. Научные интересы: компиляторные технологии, статический анализ кода.

Valery Nikolaevitch IGNATYEV – Candidate of Physical and Mathematical Sciences, Senior Researcher, ISP RAS, Senior Lecturer, Faculty of VMK MSU. Research interests: static code analysis, vulnerability search methods, machine learning.

Валерий Николаевич ИГНАТЬЕВ – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, старший преподаватель факультета ВМК МГУ. Научные интересы: статический анализ кода, поиск уязвимостей, машинное обучение.

DOI: 10.15514/ISPRAS-2020-32(3)-3

Static analyzer debugging and quality assurance approaches

M.A. Menshikov, ORCID: 0000-0002-7169-7402 <info@menshikov.org>
St Petersburg State University,
7–9, Universitetskaya nab., St. Petersburg, 199034, Russia

Abstract. Writing static analyzers is hard due to many equivalent transformations between program source, intermediate representation and large formulas in Satisfiability Modulo Theories (SMT) format. Traditional methods such as debugger usage, instrumentation, and logging make developers concentrate on specific minor issues. At the same time, each analyzer architecture imposes a unique view on how to represent the intermediate results required for debugging. Thus, error debugging remains a concern for each static analysis researcher. In this paper, our experience debugging a work-in-progress industrial static analyzer is presented. Several most effective techniques of constructive (code generation), testing (random test case generation) and logging (log fusion and visual representation) groups are presented. Code generation helps avoid issues with the copied code, we enhance it with the verification of the code usage. Goal-driven random test case generation reduces the risks of developing a tool highly biased towards specific syntax construction use cases by producing verifiable test programs with assertions. A log fusion merges module logs and sets up cross-references between them. The visual representation module shows a combined log, presents major data structures and provides health and performance reports in the form of log fingerprints. These methods are implemented on a basis of Equid, the static analysis framework for industrial applications, and are used internally for development purposes. They are presented in the paper, studied and evaluated. The main contributions include a study of failure reasons in the author's project, a set of methods, their implementations, testing results and two case studies demonstrating the usefulness of the methods.

Keywords: static analysis; debugging; goal-driven random test case generation; code generation; log file analysis; visual representation

For citation: Menshikov M.A. Static analyzer debugging and quality assurance approaches. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 33-48. DOI: 10.15514/ISPRAS–2020–32(3)–3

Подходы к отладке и обеспечению качества статического анализатора

М.А. Меньшиков, ORCID: 0000-0002-7169-7402 <info@menshikov.org>
Санкт-Петербургский государственный университет,
Россия, 199034, Санкт-Петербург, Университетская наб., д. 7–9

Abstract. Написание статических анализаторов затруднено из-за наличия множества эквивалентных преобразований между исходным кодом программы, промежуточным представлением и большими формулами в формате Satisfiability Modulo Theories (SMT). Традиционные методы, такие как использование отладчика, инструментарий и ведение журналов, заставляют разработчиков сосредотачиваться на определенных мелких проблемах. В то же время каждая архитектура анализатора навязывает уникальное представление о том, как следует представлять промежуточные результаты, необходимые для отладки. Таким образом, отладка остается проблемой для каждого исследователя статического анализа. В этой статье представлен наш опыт отладки незавершенного промышленного статического анализатора. Представлено несколько наиболее эффективных методов конструктивной (генерация кода), тестовой (генерация случайных тестовых случаев) групп, а также группы

журнализации (объединение и визуальное представление журналов). Генерация кода помогает избежать проблем с копируемым кодом, мы улучшаем его с помощью проверки использования кода. Генерация случайных тестовых наборов на основе целей снижает риски разработки инструмента, сильно смещенного в сторону конкретных вариантов использования конструкции синтаксиса, путем создания проверяемых тестовых программ с утверждениями. Слияние журналов объединяет журналы модулей и устанавливает перекрестные ссылки между ними. Модуль визуального представления показывает объединенный журнал, представляет основные структуры данных и предоставляет отчеты о работоспособности и производительности в форме отпечатков журнала. Эти методы реализованы на основе Equid, платформы статического анализа для промышленных приложений, и используются для внутренних целей. Они представлены в статье, изучены и оценены. Основные вклады включают изучение причин сбоев в авторском проекте, набор методов, их реализации, результаты тестирования и два тематических исследования, демонстрирующие полезность методов.

Ключевые слова: статический анализ; отладка; целенаправленная генерация случайных тестовых примеров; генерация кода; анализ журнальных файлов; визуальное представление

Для цитирования: Меньшиков М.А. Подходы к отладке и обеспечению качества статического анализатора. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 33-48 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–3

1. Introduction

Software engineering is ailing from quality assurance issues. Many approaches are aiming at achieving runtime error absence (reviewed in Section 2), but logical correctness is not trivially reachable even if it is guaranteed statically. This is often the case because the correctness is not automatically derived from internal consistency.

The static analysis is aimed at verification of the real-world problems written in the form of programs. Analyzers follow the generic debugging and quality assurance trends, however, there is specificity which should be taken into account [1]. The input program undergoes several transformations, and each of them has a significant impact on the validity of the final result. Moreover, several transformations, when combined, may have cumulative effects. The issues during transformations are usually *not* runtime errors, which are easy to narrow down using traditional methods, but rather logical defects. Since transformations are unique products of each static analyzer, quality assurance is the sole responsibility of the analyzer's author.

In the Equid project [2], the author had several observations. First, feature testing looked biased towards the developer's interpretation: there is a tendency to test constructions in a way they are used by the person. Second, thorough bugs may be hidden behind multiple abstractions and appear unexpectedly as analyzer grows. Third, a *significant amount* of errors could have been found if there was a simple measure indicating that action was required. Inserting heuristics for every aspect of a large software project log is barely achievable (consider limited system resources when making such advanced logging), so log analysis is the foremost goal for analysis debugging. Fourth, contracts and formal requirements undeniably contribute to the quality of the product, however, they cover integrity and consistency rather than the absence of logical issues. In that sense, static analyzers have no specificity. And the last, static analyzers have a rare environment with little to no requirements for issue reproduction. For example, reproducing the issue in network router software may require days and months just to repeat the pattern. That makes it possible to increase logging verbosity until the issue is detected, so the developer may put efforts into making logs as informative as possible. With that in mind, the paper suggests an approach for increasing the visibility of issues and/or reducing the likeliness of bugs, based on *code generation*, *log file improvements*, *goal-driven random test case generation*, and *visual representation*. These four methods make up the author's *static analysis debugging and quality assurance approach*. The study starts with the description of third-party approaches (Section 2), continues with the key issue sources identified by the author (Section 3), the method is presented in Section 4 and evaluated in Section 5.

Motivation. The debugging field concentrates on runtime issues and doesn't answer the question how to debug deep logical issues in the static analyzer. Thus, solving this problem and employing the right methods brings many practical benefits, such as improvements in stability, reduced risk of problems after implementing new features, a faster development pace.

Novelty. The suggested approach covers a significant amount of issues found in the real-world analyzer's development. The code generation stage is enhanced with the post-processing phase in which the internal use cases are loosely verified, making integration of new objects faster. The log fusion adopts hypertext-like approach, making the output more linked and indexable, allowing for better search and filtering. To the knowledge of the author, the logging had never been integrated with the hypertext. Random test case generation usually covers trivial input data or just *compilable* programs, but in this study, it produces programs with specified verification goals, which is an improvement over completely random programs. The visual representation is usually aimed at the visualization of control flow graphs, but in this paper, it is intended for data structures and internal health/performance reports.

Main contributions. Main contributions comprise the study of the issues in the Equid project, four debugging and quality assurance techniques, implementations for 3 of the mentioned platform-independent methods, the testing results, and two case studies presenting the usefulness of the method.

2. Related work

Most works are about analyzing complex systems with static analyzers rather debugging analyzers, however, there is a study of defects in static analyzers [1], which gives useful insight to the opinions of other researchers. According to the paper, visualization and handling of intermediate results is still not satisfactory for the most developers, as well as handling of data structures. While the author hadn't synchronized with this research when the development of the analyzer started, a significant match with the practical experience had been determined.

There are well-known complex tools for debugging of complex systems, e.g. GDB [3], supporting all major Central Processing Unit architectures. The LLDB [4] is an LLVM-based GDB analogue, which aims to provide reusable infrastructure. Of course, there is a number of language-specific tools, but GDB and LLDB are among the most universal ones (e.g. UndoDB [5], which is based upon GDB, can be used to debug Java and Go applications). Such debuggers provide a way to debug tools in the direction from the beginning to the end and support analyzing core files.

The reverse direction debugging (or «omniscient debugging») is covered by GDB itself; the Mozilla's RR [6], the record and replay framework; the Undo Debugger [5], which is claimed to be one of the first commercial reverse-debuggers [7]. The reverse-debugging tools are useful for runtime errors, but the majority of the defects, at least in our project, don't fall into this category, so the usefulness is limited.

Random program generation has been first shown in [8], this method then evolved into CSmith [9] framework, which had extremely successful applications to industrial compilers. An interesting result was achieved in the paper [10], in which the author tries to avoid generating dead code by using all the temporary computations for the final result. Intel has also prepared its random program generator [11] capable of triggering compiler optimization bugs, with over 140 defects found in LLVM and GCC. The research [12] covers Orange4 random program generator with an idea of equivalent transformations – which is, by the framework, similar to goal-based generation from our study, however, the generated programs are completely random without any goals set. The MicroTESK [13] project generates test programs for various microprocessors (ARM, MIPS, RISC-V and other architectures), however, it is aimed at a lower level than the tool described in our research.

The static analysis visualization is a highly specialized topic, only applicable to concrete tool developers. Still, the authors of the paper [14] explored the ways to animate the static dependence

analysis, and the result is very different from standard still visualizations. The closest generic solutions so far are brought by code visualization tools, e.g. Sourcetrail [15], CVSscan~--- code evolution visualizer [16], which are at least capable of visualizing programs. The negative part is that local intermediate representations aren't supported in such tools. Graph-based program evolution is estimated by GEVOL [17] project, which is important for tracking defects produced by specific authors and functions. Software performance evolution had been examined in [18], as a list of functions along with performance results.

The logging is well-investigated in [19], with useful insight about the usage of logging in open-source projects. The study [20] concentrates on characterizing when do developers log the information. Additionally, researchers performed the survey on how to improve the logging. Three suggestions are relevant to our study: log filtering, categorization and analysis/visualization.

The current methods for information retrieval are mostly for natural languages. For unstructured data, it implies the usage of n-grams, machine learning and other text search methods [21]. One of the examples is DeepLog [22] system, which aims to find anomalies using deep neural network model utilizing Long Short-Term Memory (LSTM). Paper [23] reconstructs control flow graph of the distributed system to find anomalies. Anomaly detection in computer systems using decision trees is performed in [24]. Those are valid methods for unstructured log analysis, however, these methods are more suitable for malware action detection on sets of third-party applications, while the study concentrates the single application development, where defects are detected by the developer. Our log handling approach is closer to classic hypertext [25].

3. Sources of issues

Throughout 2019 author had been analyzing defects for the issues in more than 1500 commits in the closed static analysis project. Key issue sources had been identified:

- *missing support for the specific syntax/intermediate representation (IR) construction in submodules;*
- *small differences in implementations for repeating parts (classes)};*
- *transformation and ordering issues.*

The developers fix such issues promptly if they are observed, but they are not trivial to find. The author had determined the following three main reasons.

- *Low visuality of the transformation passes and the development process*. The developer sees the input, the result, but intermediate transformations might be incorrect. That might lead to false testing results.
- *Unattainable cross-dependencies between modules*. The engineer creates a new feature and edits modules to integrate it, yet different parts might be broken.
- *Low quality of tests*. This is the main reason why non-trivial issues are often not found. For example, if tests verify separate handling of `if` and `switch` constructions, there might be problems in their combinations if their implementations have interchanged code.

The following ideas were evaluated to make these issues more visible.

- Automatic generation of major cross-linked data structures to avoid unattended defects.
- Making an interface for viewing log that would detect cross-references between pass logs, visualize the steps, the internal error rate. That's similar to the approach suggested by the paper [1].
- Improving test cases: make random tests that would be more representative than those written by hand.

No single solution can be engineered to solve these problems. All typical quality assurance techniques were used in the project, which is not extraordinary, considering that the ultimate goal for the static analyzer is to promote good software engineering practices. The \textit{recipes}

suggested by this paper are extensions of the typical methods, designed specifically for static analyzers. In subsection 5.5, we predict the classes of the programs which might also benefit from the approach.

Fig. 1 summarizes the findings, matches the issues with the reasons why they are not found and the solutions.

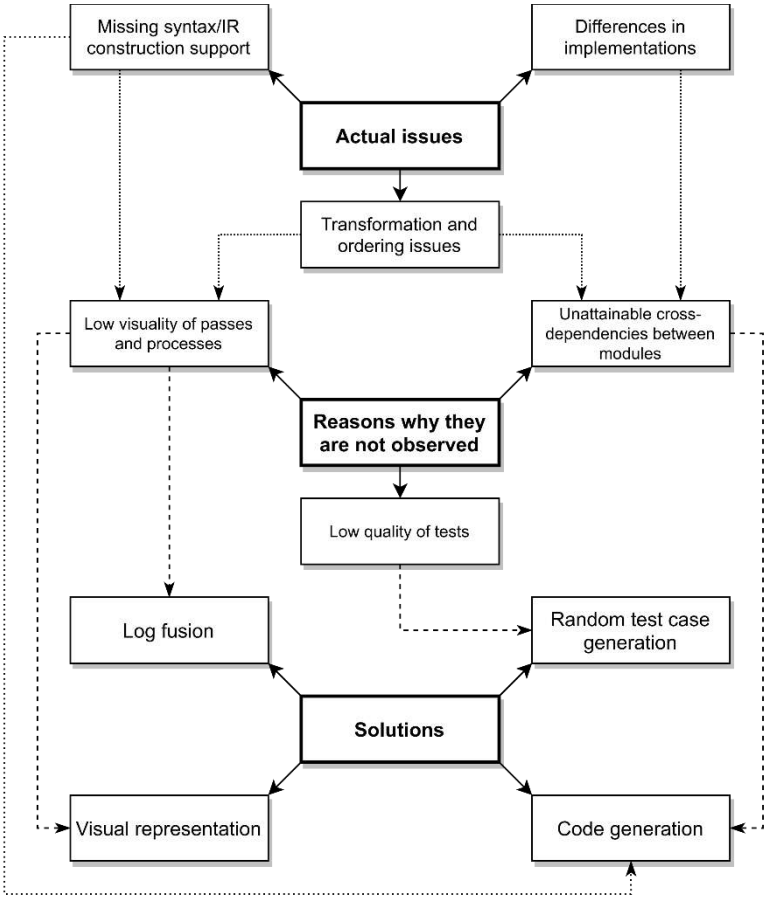


Fig. 1. Sources of the issues, reasons why these issues are not observed and their solutions

In the next section, the author's solutions to these problems are demonstrated, along with their implications.

4. Methods

As stated in the introduction, the paper aims to develop methods assisting in debugging of logical issues in static analyzers. Unlike other kinds of issues, these issues do not cause runtime failures and thus are often unattended. The form of assistance varies for every method.

4.1 Code generation

As the code base grows along with the number of syntax constructions, it gets important to ensure that repeatable fragments are written strictly and concisely, not breaking the stability of the whole program. Code generation reduces the risk of adding logical mistakes by producing modules with high integrity and compatibility. This technique itself is not new and can be applied to any software project, however, the analysis had shown that verification of the usage is vital as static analyzers

introduce transformations between different models. Also, the plurality of the models implies the need for the model instantiation for different occasions. This way, the risk of logical issues in this domain reduces due to common code generation base.

Thus, in the case of Equid, there are three major considerations for the code generation.

- **Enumerations**. When a new element is added to the enumeration, there is a high chance that dependent enumerative functions are invalidated. A mechanism that indicates the expected use of enumeration within the source code had been developed.

  For example, if a selected code needs to handle just specific enumeration elements, then, before using the enumeration `EnumerationName`, the developer may indicate the all-variant usage of the enumeration:

  ```
  core_indicate_use(EnumerationName, CoreEnumUse::AllVariants).
  ```

  If the code intentionally uses just selected enumeration values, then the developer indicates it:

  ```
  core_indicate_use(EnumerationName, CoreEnumUse::Selected).
  ```

  The post-code generation phase verifies whether all or selected elements are used. This is a cheap yet effective mechanism to ensure that all enumeration values are processed. Noteworthy is that GCC has a switch-checking approach (`-Wswitch-enum` or `-Wswitch`), which can be used with `#pragma GCC diagnostic push` for the region selection, however, GCC's approach is compiler-dependent.
- **Repetitive classes**. A big software project doing many data transformations inevitably gets many classes representing nodes participating in different analysis passes. In most cases, nodes have a similar structure. The rule of thumb is that classes that can be described declaratively should be written this way. For instance, in the project, we cover not only syntax structures but also data classes, language semantics.
- **Multi-model data**. Input data sets may be cross-linked and can be used for different purposes. There should be a way to interpret the data differently.

4.1.1 Practical implementation

To get code generation, the author had written a standalone C++ tool[1]. This version is limited regarding supported syntactical constructions, only intended for demonstration.}. The input is in customized YAML[2] format. Several models were employed: enumerations, expressions and intermediate representation commands. The tool produces a not strictly formal syntax tree, which is transformed into the real code file.

The *Enumeration* model has the following format:

```
type: <A complete type, can have reference to a different namespace
clean_type: <A type name without namespace references>
namespace: <Namespace in which enum is introduced>
dont_create_enum: [false/true]
header:
   - <extra header entry>
   - ...
field:
   - name: <Field name>
   - token: <string representation>
   - ...
unknown: <Unknown field for default alternative>
mapping:
- name: <MappingName>
```

[1] https://github:com/maximmenshikov/eq_codegen. This version is limited regarding supported syntactical constructions, only intended for demonstration.

[2] https://yaml:org/

```
- from: <Source type>
- to: <Target type>
- unknown: <default result for unmapped values>
- map:
     - from: <Source value>
     - to: <Destination value>
```

The product is a C++ `enum class` with the given name, a set of fields, a mapping function between `enum` and `std::string`, and several custom mappings.

The *Expression* model borrows many ideas from enumerations, but it is slightly more oriented towards expression model:

```
type: <Short expression type>
value_type:
   - direct: <fixed type>
   - indirect: <expression to borrow type from>
constructor:
   - name: <Internal constructor name>
   - parameter:
      - <list of parameters, named as members>
      - ...
   - ...
member:
   - name: <Friendly member name>
   - internal: <Private member name>
   - type: <Member type>
   - default: <Default value>
header:
   - <extra header entry>
   - ...
operation: <enumeration-like list of operations>
function:
   - name: <custom function name>
   - signature: <function's signature>
   - body: <function's body>
override:
   ToString: <Code that will return entry's string representation>
```

This is a short description of expression model, in fact, the model has more parameters for handling minor cases, e.g. children handling, whether the entry is LValue, and a more sophisticated return type handling.

The source for the command model (not presented in the paper) is provided at GitHub[3].

4.2 Log fusion

A typical log is a flat file with thousands of lines. Developers often struggle to find an optimal balance between verbosity and conciseness [19], but as mentioned in Section 1, static analyzers are in the unique position in which running debugging versions is possible without complete reproduction of the environment. In that case, it is possible to make tools as verbose as possible. This method does not find logical issues by itself, but, in conjunction with the fact that logs present a significant part of intermediate objects being the inputs and the outputs of transformations, it assists in making the log analysis quicker.

An approach based on the following two concepts is suggested.

1) **Module-specific logs.** Each module writes to its log, however, the core keeps track of unique timestamps for each log entry. Thus, it is possible to view separate logs if required

[3] https://github:com/maximmenshikov/eq codegen models

or to view combinations of logs. This approach has important implications from a practical perspective. First, it decreases the resources required for categorization and search. A single log would have to be parsed from the beginning to the end – which would inevitably mean a slowdown. Second, the reader gets an opportunity to select points of interest and completely avoid unrelated parts.

2) **Internal bookmarks/tags.** Each object (in our classification, a resource, a fragment or a virtual machine command) has unique ID within the object pool. Objects are referenced by a tuple (*name*:*id*:*type*), and each action on the object is bookmarked by a tuple (*id*:*type*:*action*). The log viewer allows quick navigation between these objects and by that reduces the cognitive load on log reader. This implies that the log is less of \textit{flat} structure, but rather a technical document, more oriented towards understanding.

4.2.1 Implementation

Separate logging is very implementation-specific and novelty-free. The analyzer simply opens a number of streams and provides a debug context object allowing for access to all of these streams.

The bookmark approach requires attention to the implementation of Uniform Resource Name (URN) producing methods. URN must be both readable and short, since reading log files in plain text is still an option. In the author's implementation, the URN doesn't adhere to RFC 2141[4] to save space. The practical URN grammar is presented in the corresponding GitHub repository[5].

4.3 Goal-driven random test case (program) generation

Order of actions and bias towards specific use case for syntax structures is the major source of the issues during the development. They are not detectable mainly because preparing a reasonable selection of tests proving the issue source is hard. The random test program generation is helpful in such cases due to its ability to test software against large volumes of varying input data. In result, not only logical issues are detected, but also a number of runtime issues, as seen in compilers.

A random program generator creating a set of tests with the following properties was prepared. First, all tests have one goal defined and asserted in `main()`, this is unlike other random program generators, which produce *compilable* programs without assertions. Second, all of the test cases are identical in terms of final results.

Our algorithm revolves around the idea of a \textit{verification goal}. The straight-forward algorithm for `main()` is as follows.

1) Create a function `main()` with empty block.
2) Insert a randomly named variable. Let it be *x*.
3) Assume a random goal as a target value for variable *x*. Let it be *a*.
4) Generate a random block or a random function returning *a* or modifying the input pointers so that *x* is receiving *a*.
5) Insert an assertion *x* = *a*.
   A random block or a random function is generated accordingly. The goal is transformed into a final statement (e.g. `return a` or `x = a`, based on whether the block or a function is being generated).
6) The statement is transformed into a different statement based on the random value (which chooses the next operation from the list below):
   a) if the statement isn't a block, it is transformed into a block of statements;

[4] https://tools:ietf:org/html/rfc2141
[5] https://github:com/maximmenshikov/eq-urn-grammar

b) if the statement is a block, then a new variable is added to a block;
c) if the statement is a block, then a random (not goal) variable is assigned.
d) the statement is rolled into the `if else if else` statement, where the goal is put into the first `if then` statement. For the rest of the branches, the false goal is generated and rolled into a random block.
e) the statement is rolled into the `switch` / `case` statement, where the goal is put into the first `case` body. For the rest of the cases, the false goal is generated and rolled into a random block. The false cases are randomly ended with `break`.
f) the statement is rolled into `for` or `while` loop. In the author's implementation, these constructions were of minor interest, so they were implemented trivially, similarly to Orange4 [12]} same-assignment.

After executing the first two procedures, the outcome is a valid program which must be well-parsed (syntactically correct by construction), should be analyzable and, if compiled, must satisfy all assertions. To get a set of programs, the shuffling is performed on all constructions allowing for it (the showcase is for `if` and `switch`). `if` branches are shuffled if the condition expressions are not intersecting (i.e. swapping branches doesn't change the semantics), `cases` are switched based on `break` existence. **All-break** switch-case statements can be shuffled completely.
The practical implementation is located in corresponding GitHub repository[6].

4.4 Visual representation

The visual introspection assists in finding logical issues by using a graphical view. A number of issues, especially those involving formulas and type conversions, are not distinguishable in textual forms. The following directions were in the focus of the research. First, making passes visually observable. Second, provide reasonable health reports for performance figures, error occurrence rates.

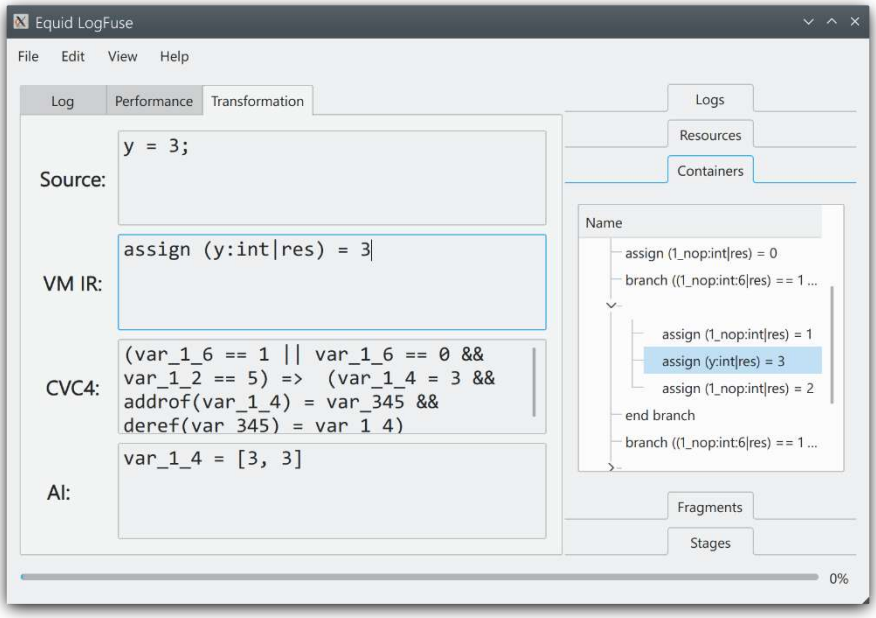


Fig. 2. The transformation view of IR commands

[6] https://github:com/maximmenshikov/eq_fuzzytest

As for the first goal, our main intention was to make an internal Virtual Machine intermediate representation (IR) viewable. The IR can be represented using a tree view since branches may contain child entries. To simplify interface development, a plain tree view from Qt7 framework had been used. Clicking on any IR command brings a list of related parts: a simplified SMT representation, a simplified abstract interpretation view (Fig. 2). At the moment of writing, the IR representing module is not directly linked to the debugger e.g. via GDB/MI interface [26], but the project's debugging framework is capable of generating the commands to set the breakpoints at the specific execution points (like IR transformation phases) for the RR [6] replaying.

As for the second, the following formulas had been used to prepare a chart. The first formula is trivial. Consider t_i, a start time of ith verification phase, where $i \in [1, n]$, and t' is the final execution time. If, for simplicity, $t' = t_n + 1$, then durations are calculated accordingly: $\Delta t_i = t_{i+1} - t_i$, where $i \in [1, n]$. However, this computation gives a very rough approximation of internal time spans.

Module log separation provides two other useful empirical formulas. One formula is based on *log size*. Consider that the logging is more or less uniformly spread around the modules. In that case, module log size is a simple profiler for the module execution times, without an actual profiler running.

The other formula requires building a time series. The complete execution time t' is collected, and it is divided up to 40 chunks: $\Delta t_i = t' / 40$, where $i \in [1, 40]$. The graph with timestamp occurrence frequencies for each Δt_i group is built for every module. The result of the implementation can be roughly described as a digital fingerprint for the execution (Fig. 3). The source code is located at GitHub[8].

Concluding, these three formulas provide insight into performance. They consider (a) total phase time, (b) total time spent in the module, (c) time distribution. They are useful if logging invocation distribution is uniform.

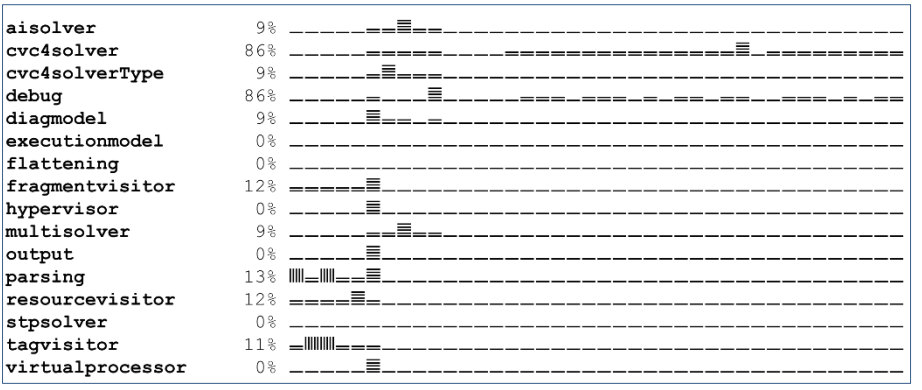


Fig. 3. The «fingerprint» of the execution

5. Evaluation

The implementation of the proposed approaches had been tested in the Equid project. For random test case generation, it is common to measure how many errors of which severity had been found using the technique, that's what determines the real usefulness of the method.

For log fusion, the developer's time to find an error was continuously monitored. This way is not accurate since the issues might differ at every testing instance.

[7] https://github:com/maximmenshikov/eq fuzzytest

[8] https://github:com/maximmenshikov/loghealth

For code generation, we investigated the time needed to bring up a new syntax structure. We were lucky to have had a syntax processing refactoring right after tool bringup due to new language requirements to the static analyzer (which are out of the scope of the paper), that made the evaluation easier.
Visual representation isn't trivially testable. The only evaluation the author could do is the subjective contribution to the issue investigation.

5.1 Random test case generation

After introducing the random test case generation, the author observed the decrease in a number of issues with both existing and newly added syntax constructions. Several defects were found, which could be classified as logical, performance, ordering and runtime issues. Logical issues comprise of verification-breaking issues, not related to ordering (which is a separate group). Performance issues are due to slow handling of syntax constructions. For this group, the time required for the execution of generated source had been evaluated and tested for sanity. The time twice larger than the empiric average for the syntax construction had been considered an error. Ordering issues appear during transformations when specific elements can't be trivially reordered in a destination form. When resulting messages differ for reordered sources, the case is considered a failure.

Table 1. Discovered issues & their severity

Defect type	*Number of issues*	*Severity*	*Comment*
Performance	3	Medium	Slow handling of specific combinations of syntax constructions, branches, especially with a high number of objects
Ordering	5	High	Ordering of syntax constructs affects the processing. This kind of issues appears during the transition from AST to IR form due to change in linearity
Runtime failure	1	High	Other critical issues with failing statements
Logical issues	1	Medium*	Problems with expression-to-formula mapping. * – This issue usually has high severity, but this concrete case was not as critical

The results of testing are provided in Table 1. In total 10 issues had been detected during the evaluation, 6 of them had high severity, and 4 of which had medium severity. The author considers the method applicable to finding mistakes in static analyzers but needs significant improvement to cover all language features. However, it is hard to judge the usefulness for compilers because no investigation had been done.

5.2 Log fusion

The logging engine can be practically evaluated only by checking the time to resolve a typical issue. The evaluation time (1 month) had been divided into two periods, in one period no logging features were used during issue-resolving, the other period is characterized by intensive usage of log fusion.
The time to resolve the typical issue reduces twice or thrice (see table 2). The improvement rates are 1.92, 5, 3.3, 1, which result in an average of 2.8 among these test groups. These results also indicate that the method is feasible for static analyzer development tasks, however, the effect is vastly different for different groups (and, supposedly, tests). But, at least, the method doesn't make the process slower.

Table 2. Time to resolve typical defects

Defect type	*Time to resolve before (h)*	*Time to resolve after (h)*	*Comment*
Performance	25	13	Performance issues require significant refactoring, but it was taking time to properly diagnose where does the issue appear
Ordering	5	1	Bookmarks make ordering issues easier to diagnose
Runtime failure	1	0.3	Runtime failures are easy to work around, but harder to fix completely. All information in one place makes it quicker
Logical issues	1	1	No large difference~--- when a logical issue is expected, you watch the log with this information. Technically, it reduces the need to find the mapping, but we haven't found it measurable

5.3 Code generation

The code generation covers a significant part of the process of adding a new syntax structure. For the project, class support had to be implemented again due to customer's requirements changed the project's infrastructure. The result is as follows. It was determined that adding class support has taken 7 times less time than the same feature several years before this test. Moreover, it was noticed that previous attempts had a month-long trail of commits revising the architectural modules and minor issues, however, the attempt after introducing code generation didn't have so many visible effects. The representability of this empirical test is very low: after all, the project has become more mature over years, however, it is hard to perform a more fair comparison to see the improvement.

5.4 Visual representation

For the visual representation, the low improvement for maintenance development phases was observed. The reason is that no developer or tester would ever look at visual reports for everyday testing. However, it is profitable for the active development phase. At least 2 performance issues were discovered using the performance chart implemented in our log viewing tool. They were related to different timings between stages, while the overall result was about the same: this situation happened due to substantially simplified processing of structures due to all of them getting the same visibility level. The deeply nested test had much longer table lookup time with much shorter propagation stage. While the visual representation testing implies little representability, the whole method can benefit if the developer is taught to have a critical look on charts.

5.5 Classes of programs

During the evaluation, the techniques had been tested not only on the main static analyzer project but also on various software packages surrounding it, to a possible extent. Author's experience shows that not only static analyzers may benefit from these methods. The class of «compatible» software comprises the programs performing a significant number of transformations. It includes the compilers, their optimization passes, refactoring, code obfuscation tools, archivers, encryption tools. The improvement would be seen in case both the input and the output are in the readable representation and if the intermediate representations are cross-linked. The approach is not cost-effective if the project is small due to a high level of an initial investment.

5.6 Case studies

5.6.1 The case of GNU statement expressions

In this case study, we would like to stress how the developed software package helps add new functionality. The GNU Compiler Collection (GCC) has the support for *statement-expressions*, which represent code blocks with the last statement being the result of the block. The infrastructure of the analyzer was highly biased towards blocks and statements, and the statement expression was an example of the construction which could be used on the unexpected levels.

By simply adding a new compound type («BlockWithResult») to the model, the code generator-related tools had shown the places which had to be touched. These areas included \textit{parsing}, *expression flattening*, *expression cloning*, *type deduction* and *IR conversion*. However, the IR conversion stage was not ready for the adoption of the statement expressions, it took around 7 working days to refactor the algorithm for it. The visualization approach let the author find the issue with the incorrect placement of internal statements: e.g. conditions were set on entering *wrong* fragments. The random program generation supported the process by providing a suitable number of examples. In total, the addition of statement expressions took approximately 10 working days.

5.6.2 The case of wrong constructors

This case is more towards mechanical mistakes when writing the code. The author did a mistake making a constructor with `std::string` parameter and a constructor with `bool` parameter. When passing regular strings, they are internally represented by `const char*` object, and the closest implicit casting for the argument was to `bool`. This mistake flowed from a Directed Acyclic Graph (DAG) level to VM intermediate representation and then materialized in missing predicate check during the verification stage. The issue had been noticed after using visual representation: it was determined that the object in DAG was missing a minor property only after reading the expression dump linked to the VM IR command. The omniscient debugging wasn't of help because the time to break the execution was unclear.

6. Conclusion and future work

In the paper, the sources of errors in the author's static analyzer project were studied. Defects are mostly related to logical issues plaguing from missing syntax/IR support, minor issues in repeating parts, transformation defects and ordering problems. To cope with them, four sustainable solutions were prepared and shown. They include random test case (program) generation, log fusion, code generation and visual representation. These methods allowed finding at least 10 defects and decreased the time to resolve defects by 2.8 on average. The response differs for different test groups or even tests, from 5x for ordering issues, down to 1x (no improvement) for logical issues. Two presented case studies support the thesis of applicability of these methods.

In future, we expect to continue improving the functionality of the logging package and increasing the number of cross-links between log parts. Code generation will experience further generalization of the models. More metrics will be investigated to make health reports more useful.

References / Список литературы

[1]. Lisa Nguyen Quang Do, Stefan Krüger, Patrick Hill, Karim Ali, Eric Bodden. Debugging static analysis. IEEE Transactions on Software Engineering, 2018.

[2]. M. Menshikov. Equid – a static analysis framework for industrial applications. Lecture Notes in Computer Science, vol. 11619, 2019, pp. 677–692.

[3]. GDB: The GNU Project Debugger. Available at: https: //www.gnu.org/software/gdb/.

[4]. The LLDB Debugger. Available at: https://lldb:llvm:org.
[5]. The interactive reverse debugger for Linux-based applications. Available at: https://undo.io/solutions/products/undodb-reverse-debugger/.
[6]. R. O'Callahan, C. Jones, N. Froyd, K. Huey, A. Noll, and N. Partush. Engineering record and replay for deployability. In Proc. of the 2017 USENIX Annual Technical Conference (USENIX ATC'17), 2017, pp. 377–389
[7]. J. Engblom. A review of reverse debugging. In Proc. of the 2012 System, Software, SoC and Silicon Debug Conference, 2012, pp. 1–6.
[8]. E. Eide and J. Regehr. Volatiles are miscompiled, and what to do about it. In Proc. of the 8th ACM International Conference on Embedded Software, 2008, pp. 255–264.
[9]. X. Yang, Y. Chen, E. Eide, and J. Regehr. Finding and understanding bugs in c compilers. In Proc. of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, 2011, pp. 283–294.
[10]. G. Barany. Liveness-driven random program generation. Lecture Notes in Computer Science, vol. 10855, 2018, pp. 112–127.
[11]. V.Yu. Livinskij, D.Yu. Babokin. Automation of search for optimization errors in C / C ++ language compilers using the Yet Another Random Program Generator. In Proc. of the 60th All-Russian Scientific Conference of MIPT. Radio engineering and computer technology, 2017, pp. 40-42 (in Russian) / В.Ю. Ливинский, Д.Ю. Бабокин. Автоматизация поиска ошибок оптимизации в компиляторах языков C/C++ с помощью генератора случайных тестов Yet Another Random Program Generator. Труды 60-й Всероссийской научной конференции МФТИ. Радиотехника и компьютерные технологии, 2017 г., стр. 40-42.
[12]. S. Takakura, M. Iwatsuji, and N. Ishiura. Extending equivalence transformation based program generator for random testing of c compilers. In Proc. of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, 2018, pp. 9–15.
[13]. M. Chupilko, A. Kamkin, A. Kotsynyak, and A. Tatarnikov. Microtesk: Specification-based tool for constructing test program generators. Lecture Notes in Computer Science, vol. 10629, 2017, pp. 217–220.
[14]. D. Binkley, M. Harman, and J. Krinke. Characterising, explaining, and exploiting the approximate nature of static analysis through animation. In Proc. of the 2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation, 2006, pp. 43–52.
[15]. Sourcetrail – documentation. Available at: https://www:sourcetrail:com/documentation.
[16]. L. Voinea, A. Telea, and J. J. Van Wijk. Cvsscan: visualization of code evolution. In Proc. of the 2005 ACM symposium on Software visualization, 2005, pp. 47–56.
[17]. C. Collberg, S. Kobourov, J. Nagra, J. Pitts, and K. Wampler. A system for graph-based visualization of the evolution of software. In Proc. of the 2003 ACM Symposium on Software Visualization, 2003, pp. 77–86.
[18]. J.P.S. Alcocer, F. Beck, and A. Bergel. Performance evolution matrix: Visualizing performance variations along software versions. In Proc. of the 2019 Working Conference on Software Visualization (VISSOFT), 2019, pp. 1–11.
[19]. D. Yuan, S. Park, and Y. Zhou. Characterizing logging practices in open-source software. In Proc. of the 2012 34th International Conference on Software Engineering (ICSE), 2012, pp. 102–112.
[20]. Q. Fu, J. Zhu, W. Hu, J.-G. Lou, R. Ding, Q. Lin, D. Zhang, and T. Xie. Where do developers log? an empirical study on logging practices in industry. In the Companion Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 24–33.
[21]. D. Jurafsky, J. Martin, P. Norvig, and S. Russell. Speech and Language Processing. Pearson Education, 2014, 1032 p.
[22]. M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proc. of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1285–1298.
[23]. A. Nandi, A. Mandal, S. Atreja, G. B. Dasgupta, and S. Bhattacharya. Anomaly detection using program control flow graph mining from execution logs. In Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 215–224.
[24]. O.I. Sheluhin, V.S. Rjabinin, and M.A. Farmakovskij. Anomaly detection in computer system by intellectual analysis of system journals. Voprosy kiberbezopasnosti, vol. 26, no. 2, 2018, pp. 33-43 (in Russian) / Шелухин О.И., Рябинин В.С., Фармаковский М.А. Обнаружение аномальных состояний компьютерных систем средствами интеллектуального анализа данных системных журналов. Вопросы кибербезопасности, том 26, no. 2, 2018 г., стр. 33-43.

[25]. B. John Smith F. Stephen Weiss. Hypertext. Communications of the ACM, vol. 31, no. 7, 1988, pp. 816–819.

[26]. R. Stallman, R. Pesch, and S. Shebs. Debugging with GDB: The GNU Source-Level Debugger. 12th Media Services, 2018, 826 p.

Information about authors / Информация об авторах

Maxim Alexandrovich MENSHIKOV – PhD student of the Department of System Programming. Research interests: static analysis of programs, debugging tools.

Максим Александрович МЕНЬШИКОВ – аспирант кафедры системного программирования. Научные интересы: статический анализ программ, средства отладки программ.

DOI: 10.15514/ISPRAS-2020-32(3)-4

Code generation for floating-point arithmetic in architecture MIPS

I.S. Arkhipov, ORCID: 0000-0002-8566-1654 <arkhipov.iv99@mail.ru>
St Petersburg State University,
7–9, Universitetskaya nab., St. Petersburg, 199034, Russia

Abstract. This article is related to code generation for floating-point arithmetics in the MIPS architecture. This work is a part of the «RuC» project. It is specialized only in code generation for operations with floatting-point numbers. This paper does not consider lexical, syntactic, and species-specific analyses.

Keywords: code generation; translator; floating-point arithmetic; MIPS

For citation: Arkhipov I.S. Code generation for floating-point arithmetic in architecture MIPS. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 49-56. DOI: 10.15514/ISPRAS–2020–32(3)–4

Генерация кодов для вещественной арифметики в архитектуре MIPS

И.С. Архипов, ORCID: 0000-0002-8566-1654 <arkhipov.iv99@mail.ru>
Санкт-Петербургский государственный университет,
Россия, 199034, Санкт-Петербург, Университетская наб., д. 7–9

Abstract. Эта статья посвящена генерации кода для вещественной арифметики в архитектуре MIPS. Эта работа является частью проекта «RuC». В ней рассматривается только генерация кодов для операций с числами с плавающей запятой. В статье не рассматриваются лексический, синтаксический и видозависимый анализы.

Ключевые слова: кодогенерация; транслятор; арифметика чисел с плавающей запятой; MIPS

Для цитирования: Архипов И.С. Генерация кодов для вещественной арифметики в архитектуре MIPS. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 49-56 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–4

1. Introduction

RISC and CISC architectures, unlike stack architectures and virtual machines systems, have different ways to express high-level language features. There are many registers for working with data, which creates a large variability in optimal code generation. Therefore, code generation in these architectures is quite a difficult task.

For work with such architectures, a technique of request and response [1] have been developed at the mathematics and mechanics faculty of the Leningrad State University: from the top of the constructed parse tree the requests for values are received, and from below the answers – form submission parse (register, memory, constant). In addition, there are certain relationship agreements. For example, in the MIPS32 architecture, function parameter values must be in some specific registers, and function values must be in other specific registers. There are stored registers that must be preserved when calling functions, and there are non-stored (unsafe) registers. For example, the left operand of a binary formula must be represented in a stored register if the right operand has calls or slicing that apply the same rules as functions, otherwise the left operand can also be represented

in an unsaved register. Determining the complexity of the right operand is the task of optimizing parse.

This work has practical application: the MIPS32 architecture is the basic architecture of one of the Russian computers Baikal-T1 [2].

The goal of this work is to generate codes for floating-point arithmetic in MIPS32 codes using the request and response technique.

2. Motivation

The development of a domestic translator is an actual task, since many industries require domestic software to avoid «back doors» in foreign software. Writing your own translator is a difficult task. This work is part of the «RuC» project [3] and is specialized only in code generation of operations with floating-point numbers. At the moment, this project has a customer, which is an additional evidence of the relevance of this work.

3. Problem statement

To achieve the goal of this work, the following tasks were set:

- to implement code generation for operations with floating-point numbers in RuC using the query and response technique;
- to implement printing floating-point numbers to the console;
- to prepare tests and test the implemented code generation.

The results can be considered successful if the assembler code received during code generation is executed on the Baikal-T1 machine and displays the correct result in the console.

4. Overview

Since this work is part of the RuC project, the same ideas as in the RuC are used to achieve the set goals. The code generator will view the parsing tree of the program and generate code based on the lexemes located in it.

It is necessary to describe the principles of RuC in general. The RuC translator has a traditional two-view structure. On the first view a scanner (lexical analyzer), a view-independent analyzer (parser) and a view-dependent analyzer work. The result of the first view is a parsing tree. This tree is input to the second code generation view, which outputs the result in MIPS32 architecture codes. This work is a part of second code generation view module, that implements operations with floatting-point numbers. More information about RuC may be found in section wiki of 'RuC' project github [9] and in the following article [10].

It is also worth mentioning a few general decisions made during the work.

- It was decided to generate commands for working with single-precision floatting-point numbers. This is due to two reasons. Firstly, at the moment there is no need for double-precision calculations on the Baikal-T1 computer. Secondly, the computer Baikal-T1 (another name BE-T1000) has 2 32-bit p5600 processor cores of the MIPS32 r5 architecture, which makes it unsuitable for double-precision computing. For example, because of the 32-bit version, you will need two commands to load a double-precision number from memory, not one.
- Implementation of using registers manually without using LLVM [8], firstly, to support RuC, and secondly, to guarantee the absence of malicious code, since due to the huge amount of code in LLVM, it is difficult to check, for example, the absence of ``back doors''.
- Processing requests of the register-to-register type only (more on this later), since there are no commands with a direct operand for floating-point values, and working with memory is represented by only two commands: load and store.

RuC has its own virtual machine, so assembly code could be generated like this: first code in virtual machine codes is generated, and then each virtual machine instruction is translated to MIPS32 assembly code. It was decided to abandon this approach because it generated large code that is difficult to optimize in the future.

5. Related Work

In the process, we also looked at the code generated by the gcc compiler [7] and compared it with our own code. Of course, the gcc compiler has already implemented many optimizations, which makes the code generated by it better. At the moment, RuC does not have any optimizations related to arithmetic operations. Optimization is the next stage of RuC development and a topic for future work.

If you compare the code generated by RuC with the non-optimized code generated by gcc, you can see that RuC uses more temporary registers than gcc for intermediate calculations. This approach is closer to a relationship agreement in mips architecture.

RuC has its own virtual machine, so it was possible to generate assembly code like this: firstly generate code in virtual machine codes, and then translate each virtual machine instruction into mips assembly code. We abandoned this approach because it generated a large code that is difficult to optimize in the future.

As an alternative approach to code generation, generation to LLVM [8] codes can be also offered. But, as it was written above, you can not guarantee that there are no «back doors» in LLVM.

As for the application, after further improvements, RuC can be used in areas where a security guarantee is required, which is why it is not possible to use foreign software products. This is the novelty of RuC – it is the first Russian translator that modifies the C language.

6. Implementation

6.1 Parse tree lexemes

As described above, the code generator views lexemes from the parse tree. We are only interested in lexemes that describe operations with floating-point numbers, namely the following:

- `TConstf` – floatting-point constant;
- `TIdenttovalf` – take the value of an identificator;
- «Unary» arithmetic operation lexemes:
  - `ASSR` – =;
  - `PLUSASSR` – +=;
  - `MINUSASSR` – -=;
  - `MULTASSR` – *=;
  - `DIVASSR` – /=;
  - `INCR` – increment;
  - `POSTINCR` – postincrement;
  - `DECR` – decrement;
  - `POSTDECR` – postdecrement;
- «Binary» arithmetic operation lexemes:
  - `LPLUSR` – +;
  - `LMINUSR` – -;
  - `LMULTR` – *;
  - `LDIVR` – /;

- Logic operation lexemes:
  - `EQEQR` – ==;
  - `NOTEQR` – !=;
  - `LLTR` – <=;
  - `LGTR` – >=;
  - `LLER` – <;
  - `LGER` – >.

The processing of each type of lexemes will be shown below.

6.2 A technique of request and response

Before describing the processing of lexemes, it is necessary to describe the technique of requests and responses. There are several types of requests, we are interested in the following:

- `BREG` – load the result in a register «breg». «breg» is a global variable in the translator, that contains the register's number;
- `BREGF` – request on the left operand, you can get an answer. Answers will be shown below;
- `BF` – free request on the right operand.

Type of request is contained in global variable «mbox».

The types of responses are:

- `AREG` – the result in a register «areg». «areg» is a global variable in the translator, that contains the register's number;
- `AMEM` – the result in memory. Global variable «adispl» contains displacement and global variable «areg» contains register»
- `CONST` – result is a constant.

Type of request is contained in global variable «manst».

6.3 TConstf

This lexeme means that a constant request was received. After this lexeme in the tree there is a constant value. Depending on the request type, we can get a register to put the constant value in «breg». If we don't get a register, the constant value is put in a temporary register `$f4` with the pseudo instruction `li.s`. It is described about floating point registers in [4]. The type of the response is `AREG`.

4.4 TIdenttovalf

This lexeme means that the value of variable must be put in register by identificator. If this is register variable, it is necessary to move it to the register «breg» when request `BREG` or `BREGF` is received. Otherwise we must put the value of this variable from memory in register «breg» or `$f4` with the instruction `lwc1` [5].

6.5 «Unary» arithmetic operation lexemes

These operations are called «unary» operations because when processing them, it is necessary to request the right operand, and the left operand is already known. The left operand may be already in register if it is register variable or in memory. If it is in memory it is necessary to put it in register. Only a register request must be issued for the right operand since there are not operations addition, subtraction, multiplication and division for floating point numbers with register and number operands. So, left and right operands must be in registers.

After this it is necessary to execute the instruction (addition, subtraction, multiplication or division). Then if variable is in memory new value of variable is saved in memory. In «areg» register of left operand is put. The type of the response is `AREG`.

It is worth noting that the division operation is performed like the rest with a single command, in contrast to the similar operation with integers.

6.5 «Binary» arithmetic operation lexemes

«Binary» operations differ from «unary» operations in that both the left and right operands must be requested before operation is executed. In contrast to the similar operation with integers for executing operations with floating point numbers left and right operands must be in registers. That's why only a register request must be issued for the left and right operands.

After getting values of left and rights operands in registers the instruction may be executed. This stage is performed as for unary operations. The type of the response is `AREG`.

6.6 Logic operation lexemes

Just like in «unary» and «binary» operations, both operands must be in registers. That's why only a register request must be issued for the left and right operands.

Unlike in similar operations with integers floating point operations change flag `FP`. Based on the logic operation, conditional transition commands are generated. If the conditional expression is complex (contains operations «and» or «or»), it is divided into simple logical expressions, the result of which is stored in the global variable in translator. When processing subsequent conditional expressions, the value of this global variable is also taken into account.

6.7 Printf

To see the results of code generation, `printf` function must be implemented. Firstly, string in data segment is generated. String is given in parse tree after `TString` lexeme. Then text segment begins again. Address of string is put in register `$a0`. After this a register request for the second operand is created. If this operand is integer or char value of this operand is put in register `$a1` and `printf` is executed. If this operand is float pointing due to mips agreements we must convert the single precision floating point number to a double precision number. After these operations `printf` is executed.

If `printf` has more than one arguments string is divided into several parts and for each part printf is executed.

7. Evaluation

After implementing code generation for operations with floating-point numbers and printing floating-point numbers tests were prepared. Tests have been prepared that demonstrate the code generation for each operation separately and for complex expressions with floating-point operations. For example, RuC translates a program in Application 1 to the assembly code in Application 2.

It is important to note that the goal is considered achieved only when the generated code is assembled successfully and is executed on the Baikal-T1. This is significant since we can think that code generation is correct but in fact it does not work. Also in such way successfulness of this work can be demonstrated.

For this purpose:

- emulator qemu [6] was installed;
- Baikal-T1 was bought;

- Baikal-T1 was connected to a laptop.

After this the prepared tests for arithmetic operations were first tested on the emulator, and then on the Baikal-T1. The tests were successful, so we can assume that the goal was achieve. You can find tests in [3] in branch mips.

8. Conclusion

This work solves the problem of code generation in MIPS32 codes of arithmetic operations with floating point numbers. Various approaches to code generation have been considered and one of them has been implemented – direct code generation.

The novelty of this work is that this work is part of the RuC project, the first Russian translator to modify the C language in favor of programming security.

In the course of the work, important results were obtained showing the applicability of the results of this work in practice. Direct code generation was implemented MIPS32 codes for floating point arithmetic operations. The generated code was successfully run on Baikal-T1.

This work has many opportunities for further research. The RuC project is not yet complete, and some C language structures are not yet implemented. Optimization of generated code is also a big area of research. Also it is necessary to implement a linker. As you can see, there are still many sources for research.

References / Список литературы

[1]. ALGOL 68. Methods of implementing. G.S. Zeitin, ed. Publishing House of Leningrad State University, 1976, 224 p. (in Russian). / Алгол 68. Методы реализации. Под редакцией Г.С. Цейтина. Изд. ЛГУ, 1976 г., 224 стр.

[2]. Baikal-T1 specifications. URL: http://www.baikalelectronics.ru/products/35/ (in Russian), accessed: 15.05.2020.

[3]. RuC project, github. URL: https://github.com/andrey-terekhov/RuC, accessed: 15.05.2020.

[4]. System V Application Binary Interface MIPS RISC Processor Supplement, 3rd Edition. Santa Cruz Operation, 1996.

[5]. MIPS Architecture for Programmers Volume II-A: The MIPS32 Instruction Set Manual. Document Number: MD00086, Revision 5.04. MIPS Tech, December 11, 2013.

[6]. QEMU official site. URL: https://www.qemu.org/, accessed:15.05.2020.

[7]. GCC official site. URL: https://gcc.gnu.org/, accessed: 15.05.2020.

[8]. LLVM official site. URL: https://llvm.org/, accessed: 15.05.2020.

[9]. RuC project github, section wiki. URL: https://github.com/andrey-terekhov/RuC/wiki (in Russian), accessed: 15.05.2020.

[10]. A.N. Terekhov, M.A. Terekhov. RuC project for education and reliable software systems development. University News, North-Caucasian region, Technical Science, issue 3, 2017, pp. 70-75 (In Russian) / А.Н. Терехов, М.А. Терехов. Проект РуСи для обучения и создания высоконадежных программных систем. Известия высших учебных заведений. Северо-Кавказский регион. Технические науки, вып. 3, 2017 г., стр. 70-75.

Application 1

```
void main()
{
   float a = 5.1, b = 6.3, c = 2.3;
   if (c > a && b < 5.3 || 5.2 >= a)
        c += (a + b) * 3.2 - 6.7 / c;
   printf("%f\n", c);
}
```

Application 2

```
        .file 1 "tests/mips/float.c"
        .section .mdebug.abi32
        .previous
        .nan legacy
        .module fp=xx
        .module nooddspreg
        .abicalls
        .option pic0
        .text
        .align 2
   .globl  main
   .ent main
   .typemain, @function
main:
        move $fp, $sp
        addi $fp, $fp, -4
        sw $ra, 0($fp)
        li $t0, 268500992
        sw $t0, -8060($gp)
        j NEXT2
        nop
FUNC2:
        addi $fp, $fp, -96
        sw $sp, 20($fp)
        move $sp, $fp
        sw $ra, 16($sp)
        li.s $f4, 5.100000
        swc1 $f4, 80($sp)
        li.s $f4, 6.300000
        swc1 $f4, 84($sp)
        li.s $f4, 2.300000
        swc1 $f4, 88($sp)
        lwc1 $f20, 88($sp)
        lwc1 $f4, 80($sp)
        c.le.s $f20, $f4
        bc1t ELSE4
        lwc1 $f20, 84($sp)
        li.s $f4, 5.300000
        c.lt.s $f20, $f4
        bc1t ELSE3
ELSE4:
        li.s $f20, 5.200000
        lwc1 $f4, 80($sp)
        c.lt.s $f20, $f4
        bc1t ELSE1
ELSE3:
        lwc1 $f20, 80($sp)
        lwc1 $f4, 84($sp)
        add.s $f20, $f20, $f4
        li.s $f4, 3.200000
        mul.s $f20, $f20, $f4
        li.s $f22, 6.700000
        lwc1 $f4, 88($sp)
        div.s $f22, $f22, $f4
        sub.s $f20, $f20, $f22
        lwc1 $f6, 88($sp)
```

```
        add.s $f4, $f6, $f20
        swc1 $f4, 88($sp)
ELSE1:
        .rdata
        .align 2
STRING1:
        .ascii "%f\n\0"
        .text
        .align 2
        lwc1 $f4, 88($sp)
        cvt.d.s $f4,$f4
        mfc1 $5,$f4
        mfhc1$6,$f4
        lui $t1, %hi(STRING1)
        addiu $a0, $t1, %lo(STRING1)
        jal printf
        nop
        j FUNCEND2
        nop
FUNCEND2:
        lw $ra, 16($sp)
        addi $fp, $sp, 96
        lw $sp, 20($sp)
        jr $ra
        nop
NEXT2:
        jal FUNC2
        nop
        lw $ra, -4($sp)
        jr $ra
        nop
        .end main
        .sizemain, .-main
```

Information about authors / Информация об авторах

Ivan Sergeevich ARKHIPOV – undergraduate student in the Department of System Programming. Research interests: MIPS architecture, compilers, code generation.

Иван Сергеевич АРХИПОВ – студент бакалавриата кафедры системного программирования. Научные интересы: архитектура MIPS, компиляторы, генерация кода.

DOI: 10.15514/ISPRAS-2020-32(3)-5

Hardware and software data processing system for research and scientific purposes based on Raspberry Pi 3 microcomputer

P.A. Pankov, ORCID: 0000-0002-4007-451X <pankov.pavel.a@gmail.com>
I.V. Nikiforov, ORCID: 0000-0003-0198-1886 <igor.nikiforovv@gmail.com>
D.F. Drobintsev, ORCID: 0000-0001-7876-3313 <drobintsev@mail.ru>
Peter the Great St.Petersburg Polytechnic University,
29, Polytechnicheskaya, St.Petersburg, 195251, Russia

Abstract. In the past ten years, rapid progress has been observed in science and technology through the development of smart mobile devices, workstations, supercomputers, smart gadgets and network servers. Increase in the number of Internet users and a multiple increase in the speed of the Internet led to the generation of a huge amount of data, which is now commonly called «big data». Given this scenario, storing and processing data on local servers or personal computers can cause a number of problems that can be solved using distributed computing, distributed data storage and distributed data transfer. There are currently several cloud service providers to solve these problems, like Amazon Web Services, Microsoft Azure, Cloudera and etc. Approaches for distributed computing are supported using powerful data processing centers (DPCs). However, traditional DPCs require expensive equipment, a large amount of energy to run and operate the system, a powerful cooling system and occupy a large area. In addition, to maintain such a system, its constant use is necessary, because its stand-by is economically disadvantageous. The article is aimed at the possibility of using a Raspberry Pi and Hadoop cluster for distributed storage and processing of «big data». Such a trip provides low power consumption, the use of limited physical space, high-speed solution to the problems of processing data. Hadoop provides the necessary modules for distributed processing of big data by deploying Map-Reduce software approaches. Data is stored using the Hadoop Distributed File System (HDFS), which provides more flexibility and greater scalability than a single computer. The proposed hardware and software data processing system based on Raspberry Pi 3 microcomputer can be used for research and scientific purposes at universities and scientific centers. Considered distributed system shows economically efficiency in comparison to traditional DPCs. The results of pilot project of Raspberry Pi cluster application are presented. A distinctive feature of this work is the use of distributed computing systems on single-board microcomputers for academic purposes for research and educational tasks of students with minimal cost and ease of creating and using the system.

Keywords: data processing; data storage; big data; cluster; supercomputer; Raspberry Pi; Hadoop

For citation: Pankov P.A., Nikiforov I.V., Drobintsev D.F. Hardware and software data processing system for research and scientific purpose based on Raspberry Pi 3 microcomputer. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 57-70. DOI: 10.15514/ISPRAS–2020–32(3)–5

Программно-аппаратный комплекс обработки данных для исследовательских и научных целей с использованием микрокомпьютера Raspberry Pi 3

П.А. Панков, ORCID: 0000-0002-4007-451X <pankov.pavel.a@gmail.com>
И.В. Никифоров, ORCID: 0000-0003-0198-1886 <igor.nikiforovv@gmail.com>
Д.Ф. Дробинцев, ORCID: 0000-0001-7876-3313 <drobintsev@mail.ru>

Санкт-Петербургский политехнический университет Петра Великого,
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29

Abstract. В последние десять лет наблюдается быстрый прогресс в науке и технологиях благодаря разработке интеллектуальных мобильных устройств, рабочих станций, суперкомпьютеров, интеллектуальных гаджетов и сетевых серверов. Увеличение числа пользователей интернета и многократное увеличение скорости интернета привело к генерации огромного количества данных, которые сейчас обычно называют «большими данными». При таком сценарии хранение и обработка данных на локальных серверах или персональных компьютерах может вызвать ряд проблем, которые могут быть решены с помощью распределенных вычислений, распределенного хранения данных и распределенной передачи данных. В настоящее время существует несколько провайдеров облачных услуг для решения этих проблем, таких как Amazon Web Services, Microsoft Azure, Cloudera и т. Д. Подходы к распределенным вычислениям поддерживаются с помощью мощных центров обработки данных (ЦОД). Однако традиционные ЦОДы требуют дорогого оборудования, большого количества энергии для работы и эксплуатации системы, мощной системы охлаждения и занимают большую площадь. Кроме того, для поддержания такой системы необходимо ее постоянное использование, поскольку ее резервирование экономически невыгодно. Целью статьи является возможность использования кластера Raspberry Pi и Hadoop для распределенного хранения и обработки «больших данных». Такое отключение обеспечивает низкое энергопотребление, использование ограниченного физического пространства, быстрое решение проблем обработки данных. Hadoop предоставляет необходимые модули для распределенной обработки больших данных путем развертывания программных подходов MapReduce. Данные хранятся с использованием распределенной файловой системы Hadoop (HDFS), которая обеспечивает большую гибкость и большую масштабируемость, чем один компьютер. Предлагаемая аппаратно-программная система обработки данных на базе микрокомпьютера Raspberry Pi 3 может быть использована для исследовательских и научных целей в университетах и научных центрах. Рассмотренная распределенная система демонстрирует экономическую эффективность по сравнению с традиционными ЦОД. Представлены результаты пилотного проекта применения кластера Raspberry Pi. Отличительной особенностью данной работы является использование распределенных вычислительных систем на одноплатных микрокомпьютерах для академических целей для исследовательских и учебных задач учащихся с минимальными затратами и простотой создания и использования системы.

Ключевые слова: обработка данных; хранение данных; большие данные; кластер; суперкомпьютер; Raspberry Pi; Hadoop

Для цитирования: Панков П.А., Никифоров И.В., Дробинцев Д.В. Программно-аппаратный комплекс обработки данных для исследовательских и научных целей с использованием микрокомпьютера Raspberry Pi 3. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 57-70 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–5

1. Introduction

Today, huge amounts of data are being generated, the source of which is social networks, meteorological organizations, corporate firms, scientific and technical institutions, web services, smart IoT devices [1], etc. Therefore, the development of tools for storing, processing and restoring information from huge volumes of data is today one of the most important issues in the research of information technology [2]. In order to meet the growing need for storage, manipulation and recovery of information, new data centers are being created.

Traditional data centers do their job well for commercial purposes, but have a number of disadvantages:

- consist of powerful hardware that is expensive;
- require a large amount of electricity to work;
- require powerful cooling;
- occupy a large area.

In addition, the use of powerful equipment provides for its continuous workload, since the operation of such systems without performing useful tasks is expensive.

Usually, big data means big sets of huge amounts of data that are difficult to work with using traditional data management tools, because of their huge size and complexity [3].

The inevitable problems of big data include the fact that the infrastructure necessary to process huge amounts of data must be created using limited resources and strictly limited processing time periods. In addition, extracting features from such data requires the use of clusters and complex data processing applications [4]. It is often necessary to work with similar data in real time.

In addition, these data centers must have capabilities such as extreme scalability, data distribution, load balancing, fault tolerance, etc. To solve these problems, Jeff Dean and Sanjay Ghemavat created the MapReduce model [5] for processing large amounts of data on large clusters.

Apache Hadoop [6, 7] is a project of the Apache Software Foundation, an open source and freely distributed set of utilities, libraries and frameworks for developing and running distributed programs running on clusters. Apache Hadoop v2.0 consists of four main modules – HDFS (Hadoop Distributed File System), Hadoop Common (a set of software libraries and utilities), YARN (Yet Another Resource Negotiator) and Hadoop MapReduce (software framework for easily writing applications which process vast amounts of data in-parallel).

Apache Hadoop is considered one of the main technologies for interacting with big data.

A single-board computer (SBC) is a universal computer that is built on one printed circuit board together with the required processor, memory, I/O ports and other functions necessary for a well-designed computer [8]. Raspberry Pi is an inexpensive and most common single board computer. An important contribution of this study is the use of the Raspberry Pi single-board computer with Hadoop clusters, which provides parallel and distributed processing with increased performance and fault tolerance.

The system under development is considered for academic purposes for research and scientific purposes. The goals also include training employees or students to work with cluster infrastructure.

In addition, it is important to provide the ability to verify the work of the developed data processing algorithms, including in enterprises, without using the capabilities of systems for production purposes.

The main goal of the project is to create a cheap solution for academic use. This is the main feature of the project, compared with the existing solutions under consideration.

Section 2 shows the related work. Section 3 shows system design. Section 4 shows the implementation process and result of the pilot project. Section 5 presents the conclusion.

2. Review of related works

First of all, let's take a look at relatives works that use single-board computers (SBC) as a main computational unit.

2.1 Single-board computers review and selection

A single board computer (SBC) is a complete, self-contained computer. The difference between SBC and traditional personal computer is that SBC is assembled on one printed circuit board, on which all the devices necessary for the functioning of the device are installed:

- CPU;
- RAM;
- video memory;
- input-output interfaces;
- etc.

This approach to the manufacture of a single-board computer allows this to make it inexpensive and compact. In addition, the device becomes even cheaper through the use of systems on a chip (SoC). On the other hand, expanding capabilities by changing the processor, increasing the amount of memory and replacing other hardware components is impossible, since most of all these components are soldered to the board. On the other hand, these features of SBC make it possible to use them as industrial computers or as computers for embedded systems.

The big advantage of SBC is that they can easily be used as a system module from many of these modules, due to the fact that all the necessary components are on the same printed circuit board. This allows quick replacement of a broken assembly. It is enough to take a new SBC, insert an SD card with an operating system into it and connect the power and Ethernet wires.

Another advantage of using SBC is the general purpose input / output interface (GPIO) ports. A GPIO is an interface for interaction between components, for example, between a microcontroller or microprocessor and various peripherals. Most often, GPIO contacts can work both input and output, with some exceptions. The presence of GPIO ports allows you to use a SBC in embedded systems to read data from various external sensors (temperature, humidity, infrared radiation, angular speeds, accelerations, voltage, current, etc.) and control external devices (LCD displays, servos, DC motors, electric drives, LEDs and LED strips, etc.)

The following single-board microcomputers were selected for consideration:

- Banana Pi M1+;
- Orange Pi PC2;
- Raspberry Pi 3 Model B+.

There are a large number of different models of single-board computers. Table 1 compares several SBC with a similar price. As the comparison criteria were selected:

- CPU (K1);
- RAM (K2);
- network access interfaces (K3);
- supported operating systems (K4);
- price (K5).

Table 1. Comparative analysis of SBCs

	Banana Pi M1+	**Orange Pi PC2**	**Raspberry Pi 3 Model B +**
K1	A20 ARM Cortex -A7 Dual-Core 1GHz	Allwinner Cortex-A53 64-bit Quad-Core 1.2 GHz	Broadcom Cortex-A53 64-bit Quad-Core 1.4GHz
K2	1 Gb DDR3	1 Gb DDR3	1 Gb LPDDR2
K3	Wi-Fi, Ethernet	Ethernet	Wi-Fi, Ethernet
K4	Android Armbian Debian Other Linux	Android Ubuntu Debian	Raspbian Ubuntu Mate Ubuntu Core Win10 IoT
K5	3000 – 4000 rub.	2900 – 3400 rub.	3000 – 4000 rub

Based on a comparative analysis, the Raspberry Pi is the optimal choice. In addition, the Raspberry Pi is the most common SBC, which makes development easier due to community support. In

addition, the availability of Raspberry Pi in many electronics stores is a significant advantage over other SBCs.

In addition to the presented single-board computers, there are also Odroid. These single-board computers have better characteristics and greater performance, but they have a higher cost, greater power consumption, greater heat dissipation and require better cooling. In addition, Odroid is less accessible and the developer community is many times smaller than the Raspberry Pi developer community.

Raspberry Pi clusters were implemented to solve some business, scientific and academic problems. The SBC Raspberry Pi offers competitive advantages: they are inexpensive, low power, and at the same time offer features similar to a simple personal computer.

2.2 Raspberry Pi clusters

Next we provide a review of several articles and projects that conduct research on the effectiveness of using Raspberry Pi based cluster.

2.2.1 Beowulf cluster

The paper [9] presents a performance benchmarking of a Raspberry Pi 2 cluster (fig. 1). The research project shows the design and construction of a high performance cluster of 12 Raspberry Pi 2 Model B single-board computers. The Raspberry Pi 2 Model B is the second-generation Raspberry Pi. It has:

- ARM Cortex-A7 CPU 900 MHz;
- 1 Gb RAM.

All of the nodes are connected over an Ethernet 100 Mbps Network in a parallel mode. Test performed using High Performance Linpack (HPL) benchmark.

As a result of their research, the authors collected cluster performance metrics in GFlops for different number of nodes and different problem sizes.

Fig. 1. Construction of Beowulf Cluster [9]

2.2.2 Iridis-Pi cluster

The project of Iridis-Pi [10] used the Raspberry Pi (one) Model B microcomputer (fig. 2). The cluster had the following characteristics:

- 64 Raspberry Pi Model B nodes;
- Broadcom BCM2835 700 MHz;
- 512 Mb RAM.

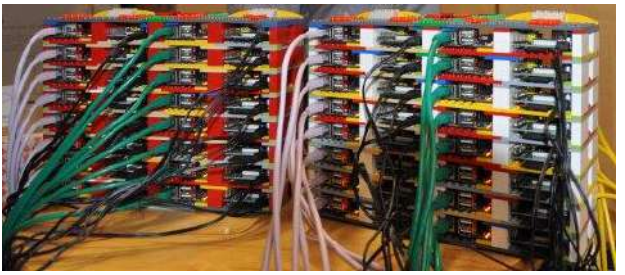


Fig. 2. Construction of Irdis Pi [12]

Like the previous project, this one uses LINPACK to test single-node performance and High-Performance LINPACK (HPL) to test cluster performance. In addition, SD card performance was measured. This is also important since the operating system and all files with which raspberry pi works are recorded on these cards.

2.3.3 Raspberry Pi Hadoop cluster and FAST algorithm

In the project [11], the Raspberry Pi Hadoop cluster is used in more realistic conditions. To test cluster performance, the authors run on it the SURF algorithm from the OpenCV library. The SURF algorithm is used to search for fixed objects (retaining their external attributes) in images using the characteristic points of the object. The similar use of the Raspberry Pi cluster in a similar task is a very illustrative example, since image processing requires high performance.

In their research, the authors compared the work of an ordinary desktop computer and a raspberry pi cluster with a different number of nodes and a different amount of data.

The result showed that the effectiveness of the built cluster occurs only with large amounts of data (in the case of the project, the required amount of data was from 64,000 and above).

2.3.4 Traditional DPC and single-board computer DPC

An important part of the research is the comparison of traditional data processing centers and data processing centers based on single-board microcomputers. As projects for comparison the last article (B) and the cluster of the higher school of software engineering of St. Petersburg Polytechnic University (A) were taken.

For comparison, the following criteria were identified:

- types of tasks to be solved (K1);
- examples of using (K2);
- hardware (K3);
- software (K4);
- hardware price (K5);
- areas of use (K6).

Comparison is presented in the Table 2.

Table 2. Traditional cluster and SBC cluster comparison

	Traditional cluster	**SBC cluster**
	A	***B***
K1	Students laboratory and course works on big data processing	Big data processing using computer vision algorithms
K2	Hadoop Distributed data processing and storage	Hadoop Image processing using the SURF algorithm

K3	- 4 Intel Xeon 6 core E5 2620 2GHz - 32 TB HDD - 256 Gb RAM	- 5 – 10 RPi 3 Model B 1.2 GHz * - 80 – 320 Gb + HDD * - 5 – 10 Gb RAM
K4	Linux Hadoop	Ubuntu Hadoop, OpenCV
K5	~ 450 000 rub.	~ 25 000 - 45 000 rub
	Traditional cluster	**SBC cluster**
	A	***B***
K6	Production Research Commercial use	Research Education Science Academic use
* depending on the number of nodes		

As we can see from the table, the advantages of using a cluster on Raspberry Pi for research and academic purposes, since it is economically more profitable. Such a system fully fulfills the functionality of a software-hardware system for distributed storage and processing of data, and high performance is not so important for research and academic purposes, unlike commercial use.

In addition, a rather important task is to create a system that is cost-effective during downtime, as traditional data centers use hardware that consumes a large amount of energy and generates a large amount of heat.

3. System description

This section describes the architecture of the project. Below will be described the hardware and software that are used.

3.1 Hardware: Raspberry Pi 3 Model B +

Raspberry Pi [12] 3 Model B + (fig. 3) is the third generation of Raspberry Pi SBCs.

Fig. 3. Raspberry Pi 3 Model B plus[1]

Raspberry Pi was originally developed as a budget platform for learning computer science, but later gained wider fame and scope.

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

3.2 Software

A cluster requires a certain set of software. First of all, nodes need an operating system. It is necessary to install a set of programs on the operating system that will allow you to create a distributed file system and perform distributed computing. Next will be described the software that was used during the research.

3.2.1 Raspbian OS

Raspbian OS [13, 14] is the main operating system for the Raspberry Pi based on the Debian Linux operating system. Raspbian was originally created by Mike Thompson and Peter Green as an independent project. The system is optimized for operation on low-performance ARM processors. PIXEL (Pi Improved Xwindows Environment, Lightweight) is used as the desktop environment.

3.2.2 Hadoop

Apache Hadoop is the open source project by the Apache Software Foundation. Hadoop is used for reliable and scalable distributed computing, but can also serve as a distributed storage of large amounts of data. Many companies use Hadoop for production and scientific purposes.

Hadoop consists of four key components:

- HDFS. Hadoop Distributed File System is a distributed file system that is responsible for storing data on a Hadoop cluster;
- MapReduce system, which is designed for computing and processing large amounts of data on a cluster;
- Hadoop Common – this module provides the tools (written in the Java language) needed on the user's operating systems (Windows, Unix, or others) to read data stored in the Hadoop file system;
- YARN module manages the resources of systems that store data and perform analysis.

HDFS. Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop. HDFS repeatedly copies data blocks and distributes these copies to the computing nodes of the cluster, thereby ensuring high reliability and speed of calculations:

- data is distributed across several machines at boot time;
- HDFS is optimized more for streaming file reads than for irregular, random reads;
- files in the HDFS system are written once and making any arbitrary entries in the files is not allowed;
- applications can read and write HDFS files directly through the Java programming interface.

MapReduce. MapReduce is a programming model and framework for writing applications designed for high-speed processing of large amounts of data on large parallel clusters of computing nodes:

- provides automatic parallelization and distribution of tasks;
- has built-in mechanisms to maintain stability and performance in case of failure of individual elements;
- provides a clean level of abstraction for programmers.

Other tools. However, Hadoop has a number of other tools. Here is some of them:

- HBase – NoSQL database that supports random read and write;
- Pig – data processing language and runtime;
- Spark – a set of tools for implementing distributed computing;
- Hive – data warehouse with SQL interface;
- ZooKeeper – storage of configuration information.

It is important that the Hadoop software allows you to use horizontal scaling. Horizontal scaling allows to reduce the execution time of the same tasks.

4. Implementation

A cluster of four Raspberry Pi nodes was created for research and a pilot project. Fig. 4 shows the assembled cluster. It was decided to use different models of the Raspberry Pi single-board microcomputer, thereby creating a heterogeneous cluster.

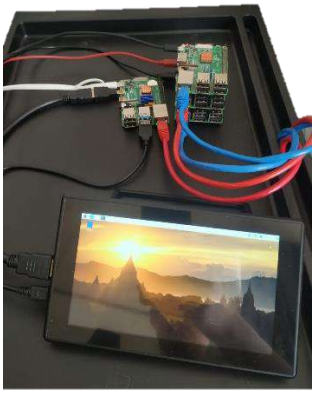


Fig. 4. Example of an assembled system of four nodes

On the created cluster, the word count algorithm using Hadoop MapReduce was tested. The Hadoop license file was used as a test file.

Table 3 shows some collected metrics from the test bench.

Table 3. Hadoop cluster performance metrics

Metric name	Value
HDFS number of bytes read	147239
HDFS number of bytes written	34796
HDFS number of read operations	8
HDFS number of write operations	2
Total time spent by map tasks (ms)	66853
Total time spent by reduce tasks (ms)	21310
Map input / output records	2746 / 21463
Combine input / output records	21463 / 2965
Reduce input / output records	2965 / 2965

The presented metrics were obtained for the operation of the cluster from one node, the second node did not participate in data processing due to configuration settings.

Next, we collected the time metrics for the algorithm for counting words in the text with different system configurations.

Tests were carried out in several stages. The algorithm was launched taking into account the fact that the text file was not divided into blocks. Further, the file system configuration was configured so that the file was divided into 3 blocks, the work with which was distributed across different nodes. Various configurations of the operating modes of the nodes were also tested. Three single-board computers performed a different role at each stage. At each stage there was 1 «master node» – it is engaged in the distribution of tasks and monitoring nodes. In addition, the number of «work nodes» that are responsible for data processing has changed. At stages 1, 3 and 5, the “master node” was at the same time a «working node».

Table 4 shows the processing time for a 38.5-megabyte file with various system configurations.

Table 4. The collected metrics of the test algorithm for calculating words in the text

	1 block	**3 blocks**
1 worker	3min 37sec	---
1 master 1 worker	2min 38sec	2min 13sec
2 workers	2min 44sec	2min 15sec
1 master 2 workers	2min 46sec	2min 14sec
3 workers	2min 49sec	2min 13sec

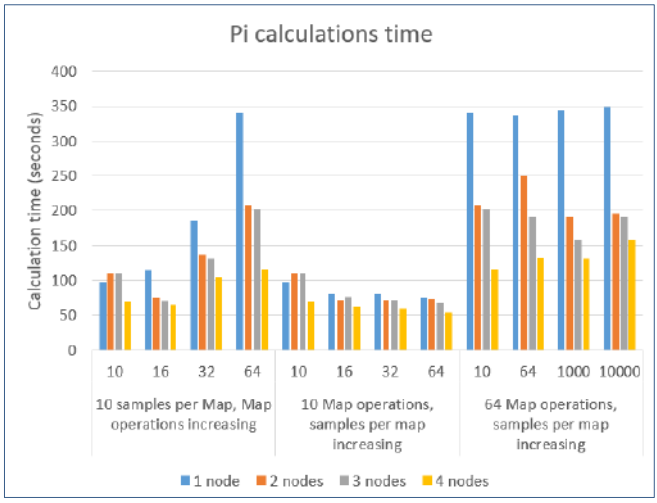
As can be seen from the table, when working on one node, the time of work with one block is the greatest. The best time was shown by the configuration of one «master node» and one «work node». In other conditions, since the file is not divided into blocks, we only lose time on the work of the «master node» for the distribution of tasks. In the case of splitting the file into 3 blocks, the execution time is approximately the same, since in each case all 3 blocks were immediately distributed to all nodes, however, the reduction in the operating time of the algorithm compared to 1 block is visible.

The second basic algorithm for checking the operation of distributed computing systems is distributed computing the value of Pi. Table 5 shows parameters of Pi calculation tests.

Table 5. Test Parameters

Constant parameters	**Variable parameters**
10 samples per Map operation	10
	16
	32
	64
10 Map operations	10
	16
	32
	64
64 Map operations	10
	64
	1000
	10000

The first results will present a graph of the dependence of the execution time of calculations on a different number of nodes and various settings that are indicated in the previous table. The graph is shown in fig. 5.



Fig, 5. Pi calculation time

In addition, a comparison was made of the time spent on one Map operation. The results are presented in fig. 6.

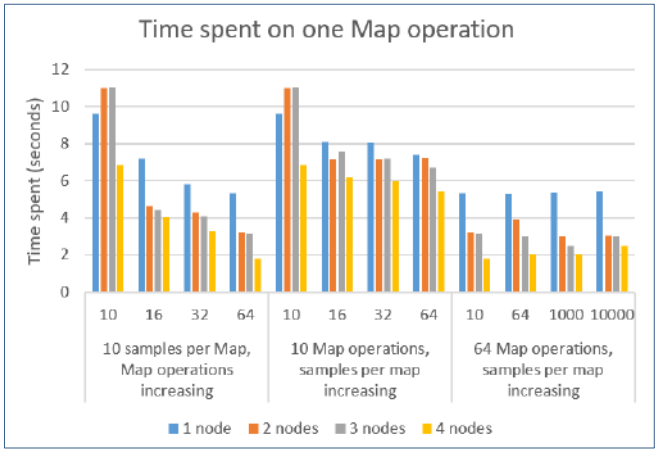


Fig. 6. Time spent on one Map operations

A distinctive feature of this work is the use of distributed computing systems on single-board microcomputers for academic purposes for research and educational tasks of students with minimal cost and ease of creating and using the system.

In addition, as a result of the study, a number of features and disadvantages of using the Raspberry Pi were identified.

- Such a cluster is effective in performing lightweight tasks, for which there is the possibility of splitting them into a large number of small tasks that do not require large computing power. The same feature leads to the fact that this cluster is not effective in such tasks where high performance is needed, for example, when working with graphics.
- SD card. Since the operating system is on an SD card, this can be a vulnerability, since the SD cards do not differ in high performance;
- During the tests, it was noticed that during prolonged operation of the cluster, the number of errors that occur during the execution of tasks increases, which may be associated with the accumulation

of garbage files. To restore stability, a reboot of the cluster was required. This leads to the need for a more detailed approach to cluster configuration.

- In a situation when the number of nodes increases, errors may occur due to the fact that the node does not have enough memory to allocate resources that the main node requests from the node. The solution to this problem is the addition of certain configuration parameters to the files of the main node. It is necessary to indicate to the main node about the need to check the availability of the requested both virtual and hardware resources. In addition, you should correctly configure the operating system itself on each node, including the amount of allocated memory for Java, which may affect the operation of the cluster. It is necessary to approach in detail the configuration of various nodes that differ in hardware characteristics.
- For comfortable operation, the Raspberry Pi requires active cooling. However, a single fan is sufficient to cool two SBC's. The fan used was powered from 5 volts and consumed 0.06 amperes, which equals a power of 0.3 watts, which is energy efficient.

5. Conclusion

As a result of the research, the following tasks were completed:

- research and analysis of existing projects on the use of SBCs within the cluster;
- comparison of the cost-effectiveness of a traditional data center and data center using SBCs for research and academic purposes. Based on this comparison, the relevance of developing a system on SBCs was revealed;
- comparison of SBCs. As a result of the comparison, the Raspberry Pi 3 Model B + single-board computer was chosen for the project, since it is the most optimal, due to its characteristics, price, and also availability and prevalence.

From the results of the test benchmarks it can be seen that the created system supports horizontal scalability, which meets the system requirements. In addition, based on the results, it can be concluded that the goal of creating a cheap, scalable distributed computing system for academic purposes has been achieved.

5. Future work

The project is under development. In the future, it is planned to perform the following tasks:

- increase the number of nodes for analysis to increase productivity;
- increase the number of metrics;
- analyze the performance of SD cards from different manufacturers, as their characteristics affect the operation of the Raspberry Pi;
- use an external USB (flash / HDD / SSD) drive (s) as storage media;
- analyze the efficiency of using a cluster on single-board computers in various tasks;
- use software tools for testing distributed applications and systems [15].

References / Список литературы

[1]. P. Pankov, I. Nikiforov, Y. Zhang. Hardware and software system for collection, storage and visualization meteorological data from a weather stand. In Proc. of the International Scientific Conference on Telecommunications, Computing and Control (TELECCON-2019), 2019 (in printing).

[2]. A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, 2008, pp. 68-73.

[3]. P. Zikopoulos, C. Eaton, D. deRoos, T. Deutsch, and G. Lapis, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill, 2011, 176 p.

[4]. L. Xue, J. Ni, Y. Li, and J. Shen. Provable data transfer from provable data possession and deletion in cloud storage. Computer Standards & Interfaces, vol. 54, 2017, pp. 46-54.

[5]. J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. in Proc. of the 6th Symposium on Operating System Design and Implementation (OSDI), 2004, pp. 137-150.

[6]. C. Lam. Introducing Hadoop. In Hadoop in Action. Manning Publications, 2011, pp. 3-20.
[7]. Tom White. Hadoop: The Definition Guide. 4th Edition, O'Rilley Media Inc., 2015, 688 p.
[8]. W. P. Birmingham and D. P. Siewiorek. MICON: a knowledge based single board computer designer. In Proc, of the 21st Conference on Design Automation, 1984, pp. 565-571.
[9]. Dimitrios Papakyriakou, Dimitra Kottou and Ioannis Kostouros. Benchmarking Raspberry Pi 2 Beowulf Cluster. International Journal of Computer Applications, vol. 179, no. 32, 2018, pp. 21-27.
[10]. Simon J. Cox et al. Irdis-Pi: A low-cost, compact demonstration cluster. Cluster Computing, vol. 17, no. 2, 2013, pp. 349–358.
[11]. Kathiravan Srinivasan et al. An Efficient Implementation of Mobile Raspberry Pi Hadoop Clusters for Robust and Augment Computing Performance. Journal of Information Processing Systems, vol.14, no. 4, 2018, pp. 989-1009.
[12]. Molloy Derek. Exploring Raspberry Pi. Interfacing to the Real World with Embedded Linux. Wiley, 2016, 720 p.
[13]. William Harrington. Learning Raspbian. Packt Publishing, 2015, 154 p.
[14]. Roberto Morabito. Virtualization on Internet of Things Edge Devices with Container Technologies: a Performance Evaluation. IEEE Access, vol. 5, 2017, pp. 8835-8850.
[15]. Kobyshev K.S., Nikiforov I.V., Prokofyev O.V. Tool for automating testing components of a distributed application using an esb bus. In Proc. of the Scientific and Practical Conference on Modern Technologies in Theory and Practice of Programming, 2019, pp. 212-214 (in Russian) / Кобышев К.С., Никифоров И.В., Прокофьев О.В. Инструмент для автоматизации тестирования компонент распределенного приложения, использующего ESB-шину. Труды научно-практической конференции «Современные технологии в теории и практике программирования», 2019 г., стр. 212-214.

Information about authors / Информация об авторах

Pavel Aleksandrovich PANKOV is a graduate student at the Higher School of Software Engineering at the Institute of Computer Science and Technology. Research interests: big data, data analysis, computer vision, robotics, automation of technological processes, embedded systems.

Павел Александрович ПАНКОВ – студент магистратуры Высшей школы программной инженерии Института компьютерных наук и технологий. Сферы научных интересов: большие данные, анализ данных, компьютерное зрение, робототехника, автоматизация технологических процессов, встраиваемые системы.

Igor Valerievich NIKIFOROV – candidate of technical sciences, associate professor of the Higher School of Software Engineering at the Institute of Computer Science and Technology. Research interests: parallel data processing, distributed data storage systems, big data, software verification, test automation.

Игорь Валерьевич НИКИФОРОВ – кандидат технических наук, доцент Высшей школы программной инженерии Института компьютерных наук и технологий. Сферы научных интересов: параллельная обработка данных, системы распределенного хранения данных, большие данные, верификация программного обеспечения, автоматизация тестирования.

Dmitry Fedorovich DROBINTSEV is a senior lecturer at the Higher School of Software Engineering at the Institute of Computer Science and Technology. Research interests: big data, information analytics, software verification.

Дмитрий Фёдорович ДРОБИНЦЕВ – старший преподаватель Высшей школы программной инженерии Института компьютерных наук и технологий. Сферы научных интересов: большие данные, аналитика информации, верификация программного обеспечения.

DOI: 10.15514/ISPRAS-2020-32(3)-6

Tracing Network Packets in the Linux Kernel using eBPF

M.G. Kovalev, ORCID: 0000-0003-1050-052X <restonich@gmail.com>
St Petersburg State University,
7–9, Universitetskaya nab., St. Petersburg, 199034, Russia

Abstract. During the development and maintenance of complex network infrastructure for a big project, developers face a lot of problems. Although there exist plenty of tools and software that helps to troubleshoot such problems, their functionality is limited by the API that Linux kernel provides. Usually, they are narrowly targeted on solving one problem and cannot show a system-wide network stack view, which could be helpful in finding the source of the malfunction. This situation could be changed with the appearance of a new type of tools powered by the Linux kernel's eBPF technology, which provides a flexible and powerful way to run a userspace code inside the kernel. In this paper, an approach to tracing the path of network packets in the Linux kernel using eBPF is described.

Keywords: Linux; kernel; networking; tracing; eBPF

For citation: Kovalev M.G. Tracing Network Packets in the Linux Kernel using eBPF. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020. pp. 71-78. DOI: 10.15514/ISPRAS–2020–32(3)–6

Трассировка сетевых пакетов в ядре Linux с использованием eBPF

М.Г. Ковалев, ORCID: 0000-0003-1050-052X <restonich@gmail.com>
Санкт-Петербургский государственный университет,
Россия, 199034, Санкт-Петербург, Университетская наб., д. 7–9

Аннотация. При разработке и обслуживании комплексных сетевых инфраструктур в больших проектах разработчики сталкиваются с множеством проблем. Несмотря на то, что существует множество инструментов для поиска и устранения таких проблем, их функциональность ограничена программным интерфейсом, предоставляемым ядром Linux. Обычно они специализируются на решении конкретных задач и не могут дать широкий взгляд на весь сетевой стек системы, что могло бы помочь в процессе поиска источника неполадки. Эта ситуация может измениться с появлением нового типа инструментов, использующих технологию ядра Linux eBPF, дающую гибкий и мощный способ запускать пользовательский код в пространстве ядра. В этой статье описывается подход к трассировке сетевых пакетов в ядре Linux с помощью eBPF.

Ключевые слова: Linux; ядро; сети; трассировка; eBPF

Для цитирования: Ковалев М.Г. Трассировка сетевых пакетов в ядре Linux с использованием eBPF. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 71-78 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–6

1. Introduction

Software and hardware solutions are becoming increasingly complex, which leads to an increasingly complex network infrastructure that lies at the basis of such solutions. Such infrastructures could

include multiple physical devices and virtual interfaces, various network namespaces, firewall settings, routing tables, packet filtering, networking protocols, and so on. These technologies are very powerful and feature-rich, but on the other hand, it also makes troubleshooting of such network systems much harder and more time-consuming.

There are a lot of ways to troubleshoot networking problems. One of them is to walk through the OSI stack: go all the way up from the link layer to upper ones checking the system to work correctly on each layer. Check if the network interface is working and is configured right, look at the ARP and routing tables, firewall rules, packet filtering, and move on to check the high-level configurations. Most problems are solved in one of these steps. But complexity of the system configuration will eventually lead to non-obvious relations between different parts of it and more difficult problems will appear. Such problems are solved by excluding possible sources of malfunction one by one with different tools. This is a long and tedious process, and it needs to be repeated for every problem over and over since, for each problem, possible sources of malfunction are new and need to be rechecked. Most of the tools, being narrowly targeted on solving specific issues, do not help either. Though doing their job very well, they cannot provide a system-wide network stack view, which could help us solve non-obvious problems in complex network infrastructures.

In this paper, the «system-wide network stack view» means a path of the network packet through Linux's networking stack. It shows which functions processed the packet and for how long, where it was consumed or dropped, or if it went the wrong way, not intended by the network architecture. With this information, the developer could narrow down the scope of troubleshooting and solve the problem quickly with the use of the appropriate tools.

In the past, information about the packet's path would not be possible to obtain without direct kernel code modification or some serious restrictions. Now it can be easily done with the use of eBPF technology.

2. Technology Overview

2.1 BPF

Berkley Packet Filter (BPF) is a technology that consists of the register-based virtual machine and the instruction set for that machine. It was designed for a highly optimized and performant network traffic filtering [1]. It is used as the backend for the libpcap library and does packet filtering for tools such as tcpdump. When tcpdump is executed with some filtering rule, it generates BPF bytecode for that rule and sends it to the kernel to attach at the early stages of the network stack processing. That bytecode then gets interpreted on the virtual machine and decides which packet shows up in tcpdump's output [2].

This filtering mechanism is performant and secure by design. BPF programs executed isolated on the in-kernel virtual machine. They are limited to 4096 instructions, cannot have loops, and all memory accesses are checked for a valid range. So, execution of the BPF bytecode is guaranteed to terminate; it cannot cause a kernel fault, a denial of service, or memory damage.

While being a useful concept of securely running userspace code in the kernel, BPF is limited by its design and age. Two registers are not enough to write powerful programs, and the instruction set is outdated as modern processors moved to 64-bit architecture. So, to take advantage of contemporary hardware, BPF had to be improved [3].

2.2 eBPF

Massive rework of the BPF was initiated in 2014 by Alexei Starovoitov [4][5].

1. 512 bytes multi-use stack space replaced old spill-fill stack.
2. The number of registers was increased from two to ten, and their width became 64-bit. All of them map one to one to hardware registers.

3. Various old instructions were modified, and new ones added, all of them becoming a close match to the hardware instructions. It greatly improved JIT compilation.
4. Maps were introduced — generic key-value data structures to exchange information between the BPF programs and with the userspace.
5. New attachment points for the programs were implemented: kernel probes (kprobes), tracepoints, perf events, and sockets. These programs are invoked every time an attachment point is passed by, and they have access to the corresponding context.

These improvements significantly increased the programmability and performance of BPF. After some other modifications, API of that rework was frozen and named as extended BPF (eBPF) [6]. Since then, eBPF has been actively developed, providing more usability and flexibility for extending the Linux kernel's functionality without editing its source code. There is an example of the Linux TCP stack extension from the user space with the help of eBPF [7].

Plenty of attachment points makes eBPF a very useful technology for creating tracing and profiling tools. bcctools and bpftrace are great examples that make use of this functionality. A thorough explanation of how to use these tools in performance testing can be found in Brendan Gregg's «BPF Performance Tools» [8].

2.3 Toolchain

Classic BPF programs were written directly in VM instructions. This approach would be restricting for the eBPF as it would be harder to use new features and extensions. To simplify programming, the eBPF backend for LLVM was introduced [9]. It allows to write eBPF programs in restricted C language and then compile them to the ELF objects with a clang. Restrictions for C language come from the eBPF design and features of ELF parsing.

1. Main program functions and map structures have to be defined with section() attribute as loaders need eBPF objects to be self-contained in the ELF sections.
2. Multiple programs can be described inside a single C file in different sections.
3. No global variables, constant strings, or arrays allowed.
4. The stack is limited to 512 bytes.
5. No ability to call library functions (except for those defined with inline in included headers or for eBPF helpers).
6. Only bounded loops are available.

These are not all of the limitations and features of writing eBPF programs in C. An up-to-date list with explanations can be found in Cilium's BPF and XDP Reference Guide [10]. Since technology is being actively developed, some restrictions are being fixed. For example, bounded loops were introduced relatively recently, and before that, no loops at all were allowed in the eBPF programs (or they had to be unrolled with pragma directive) [11].

The eBPF helpers are special in-kernel functions intended to expand functionality and ease programming. They are used to interact with the eBPF maps, get the time elapsed since system boot, print some information for debugging, edit the network packets, and many more. The exact set of helpers accessible by the eBPF program is determined by its type [12].

Successfully compiled ELF objects with eBPF objects (programs and maps) are passed to the kernel via loader. This is done via bpf() syscall, but for ease of use, bpftool loader backed by libbpf library should be used instead. Though bpftool covers overall management of eBPF objects, the attachment eBPF programs to the network path should be done via the tc tool from the iproute2 suite.

When the eBPF program is loaded into the kernel, it is processed by a static verifier. A directed acyclic graph is created from the program to check for the loops and unreachable instructions. Then the verifier simulates the execution of the program for every possible path and observes the state change of registers and stack. If the program passes, a descriptor is created for it that is then used to attach it to an appropriate attachment point.

3. Implementation

The tool has a command-line interface, taking a filter expression as an input and passing network packet path in plain text format as an output. The general sequence is shown in Fig. 1.

1. Filter expression describing network traffic that needs to be observed is passed to the tool.
2. eBPF program for the network path traffic control attachment point (TC program) is generated from that filter and loaded into the kernel along with skb_map and path_map eBPF maps.
3. eBPF programs for the attachment to kprobes of the main network functions (kprobe programs) are compiled and loaded into the kernel.
4. The TC program matches every packet against the filter. Upon finding a match, the program stores a pointer to the packet's sk_buff structure in the skb_map.
5. The kprobe program checks arguments passed as the probed function's context, and if it matches the pointer stored in the skb_map, program stores current timestamp in the path_map. If the program's probed function is marked as the last in packet path, it also fills the skb_map with 1's. That serves as a signal for the tool to stop the packet tracing.
6. The tool retrieves information from the path_map, sorts it by the timestamp values and passes it as the output.

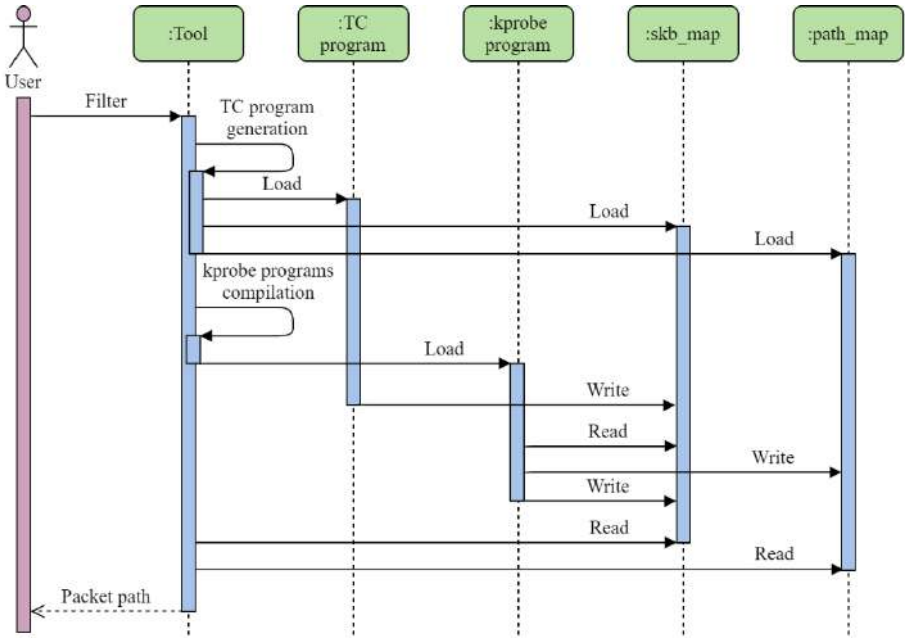


Fig. 1. Program flow implementation details

Every eBPF program has a different context passed to it based on the program type (which depends on the attachment point). For the TC programs, it is a struct `__sk_buff *skb` and for the kprobe programs – `struct pt_regs *ctx`.

struct `__sk_buff` is a user-accessible mirror of the in-kernel struct `sk_buff`. It is not a copy, more like access instructions. Accesses to the fields of this struct are processed in the eBPF verifier and translated to the accesses to the same fields of the real buffer structure. This approach improves security and portability, as programs do not rely on the in-kernel definition of the `sk_buff`.

struct `pt_regs` stores saved registers of the probed function. Arguments of the function are accessed from this structure by architecture-dependent macros, which improves portability of the kprobe functions.

Typically, eBPF programs cannot store their state. Therefore, eBPF maps are used to save the observed packet's pointer and to collect the information about its path. This allows to filter the packet only once in the early stages of its processing and to make the kprobe programs as simple as possible.

The TC program is attached to the appropriate point with the `tc` tool. `clsact` qdisc is added to the network interface, through which the observed traffic will be passing, and the TC program is passed to it as a classifier. Full command reference can be found in the `tc` man page [13].

To load and attach the kprobe programs, I've implemented a program based on the `libbpf` library, as `bpftool` lacks such functionality. My loader also replaces eBPF maps in loaded objects for those already created by the TC program load. This is necessary to establish communication between programs.

The Code of the kprobe programs is basically the same and simple. To attach them to the various kernel functions they are identified by the macros:

- `KP_NUM` stores a number of the probed function;
- `KP_SEC` stores a name of the probed function in the format "`kprobe/<function_name>`";
- `KP_FIN` stores 1 if probed function marked as the last one and 0 otherwise.

These values are taken from a `kp_funcs.txt` file and are filled on the compilation phase with the use of the clang's `-D<macroname>=<value>` option. This way kprobe programs for all the observed functions are created from the only one source file. This mechanism creates unnecessary overhead and is to be changed for a more suitable solution.

`kp_funcs.txt` file used in the kprobe program's compilation is necessary to provide portability for the tool. It is to be adapted for different Linux versions as names of the in-kernel functions change from time to time. Also, users can easily add new functions to this list to observe if the packets pass through them.

On the Linux systems `/etc/security/limits.conf` file controls limits of the various system values such as maximum file size, stack size, or processes count [14]. For my tool memlock value is the most important one. It is a maximum locked-in-memory address space that limits an amount of the memory pages in RAM that are not to be placed in the swap space. So, to be able to load a large amount of kprobe programs, this limit needs to be increased by the user.

4. Similar approaches

4.1 VMware Traceflow

Traceflow is a part of the VMware NSX Data Center for vSphere platform [15]. It injects packets into the network and traces them as they travel between nodes. It provides a good overview of the whole network, from which an administrator could get information about possible sources of malfunction or performance reduction.

Traceflow operates on a high level of networking and does not tell about internal packet processing. My tool can operate only on one node yet provides a thorough network packet path through the kernel. Additionally, it does not inject special traffic into the system and works on the existing one. Also, usage of the Traceflow is restricted to the vSphere platform, while my solution runs on any system with a recent enough Linux kernel version.

4.2 ftrace

ftrace system [16] also could be used to trace network packets in the Linux. This could be done by restricting all network traffic in the system except for the one that is to be observed. Then `function_graph` tracer can be used on the function such as `__netif_receive_skb_list_core()` to show the path of the incoming packet.

It is an easy and detailed approach, though not so useful on a production system. My tool traces determined network packets, and the presence of another traffic in the system does not interfere with it.

4.3 tcpdrop

tcpdrop tool is a part of the BCC (BPF Compiler Collection) project and is built on the eBPF [17]. It provides a stack trace of Linux kernel functions that led to the drop of the TCP packet along with other useful information. This helps answer why such drops are happening.

Though being limited to only TCP traffic and not observing the full path of the packets, tcpdrop shows a good example of the usefulness of the approach that lies in the foundation of my tool.

6. Future Work

The current state of the tool is as follows.

1. The TC program is static and can trace incoming ICMP, TCP, or UDP packets with a manual macro modification.
2. The kprobe programs are compiled manually with appropriate values passed.
3. Information of the packet passing through functions is printed by programs via `bpf_trace_printk()` helper and is observed through the `/sys/kernel/debug/tracing/trace_pipe` file.

To reach an MVP (minimum viable product) state for the tool, the following things are to be implemented.

1. The generation of the TC program based on the filter passed.
2. A `kp_func.txt` file composition and a kprobe programs compilation automatization.
3. A `path_map` information retrieval mechanism and a packet path composition.

At the scope of the whole project, there are several points of consideration.

1. Amount of the kprobe programs that is acceptable to sustain a balance between performance and usability of the tracing information.
2. Different use-cases need to be described as well as scenarios of troubleshooting comparison with and without this tool.
3. Kprobes are not a part of the stable Linux API, the name of the functions could change. This should be handled to guarantee the operation of this tool on various kernel versions. Also, it is possible that thorough kprobe tracing is unnecessary in some scenarios, so instead the tool could rely on the tracepoints as a more stable kernel API.

The tool source files can be found in my GitHub repository [18].

References / Список литературы

[1]. Steven McCanne, Van Jacobson. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In Proc. of the USENIX Winter 1993 Conference, 1993, pp. 259-270.

[2]. Marek Majkowski. BPF - the forgotten bytecode. The Cloudflare Blog, May 2014, available at: https://blog.cloudflare.com/bpf-the-forgotten-bytecode/.

[3]. Matt Fleming. A thorough introduction to eBPF. LWN.net, December 2017, available at: https://lwn.net/Articles/740157/.

[4]. Alexei Starovoitov. net: filter: rework/optimize internal BPF interpreter's instruction set. index: kernel/git/torvalds/linux.git, March 2014, available at: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=bd4cf0ed331a275e9bf5a49e6d0fd55dffc551b8.

[5]. Jay Schulist, Daniel Borkmann, Alexei Starovoitov. Linux Socket Filtering aka Berkeley Packet Filter (BPF). Linux in-kernel documentation, available at: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/networking/filter.txt

[6]. Alexei Starovoitov. net: filter: split filter.h and expose eBPF to user space. kernel/git/torvalds/linux.git, September 2014, available at: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=daedfb22451dd02b35c0549566cbb7cc06bdd53b
[7]. Viet-Hoang Tran, Olivier Bonaventure. Making the Linux TCP stack more extensible with eBPF. In Proc. of the Netdev 0x13, Technical Conference on Linux Networking, 2019, available at: https://netdevconf.info/0x13/session.html?talk-tcp-ebpf.
[8]. Brendan Gregg. BPF Performance Tools. Addison-Wesley Professional, 2019, 880 p.
[9]. Alexei Starovoitov. BPF backend. LLVM project, commit, December 2014, available at: https://reviews.llvm.org/D6494.
[10]. BPF and XDP Reference Guide. Cilium, available at: https://docs.cilium.io/en/latest/bpf/.
[11]. Daniel Borkmann, Alexei Starovoitov. Merge branch 'bpf-bounded-loops', kernel/git/torvalds/linux.git, June 2019, available at: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=94079b64255fe40b9b53fd2e4081f68b9b14f54a.
[12]. BPF-HELPERS - list of eBPF helper functions, manual page, available at: http://man7.org/linux/man-pages/man7/bpf-helpers.7.html.
[13]. Bert Hubert. tc - show / manipulate traffic control settings. manual page, available at: http://man7.org/linux/man-pages/man8/tc.8.html.
[14]. Cristian Gafton. limits.conf - configuration file for the pam_limits module. available at: http://man7.org/linux/man-pages/man5/limits.conf.5.html.
[15]. VMware Docs. VMware NSX Data Center for vSphere Documentation. Traceflow documentation, May 2019, available at: https://docs.vmware.com/en/VMware-NSX-Data-Center-for-vSphere/6.4/com.vmware.nsx.admin.doc/GUID-233EB2CE-4B8A-474C-897A-AA1482DBBF3D.html.
[16]. .ftrace - Function Tracer. Linux in-kernel documentation. available at: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/trace/ftrace.rst.
[17]. Brendan Gregg. Linux bcc/eBPF tcpdrop. Brendan Gregg's Blog, May 2018, available at: http://www.brendangregg.com/blog/2018-05-31/linux-tcpdrop.html.
[18]. Mark Kovalev. Bpfpath. GitHub repository, available at: https://github.com/restonich/bpfpath.

Information about authors / Информация об авторах

Mark Germanovitch KOVALEV – student of the Department of System Programming, Faculty of Mathematics and Mechanics. Research interests: network infrastructure management, operating systems.

Марк Германович КОВАЛЕВ – студент кафедры системного программирования математико-механического факультете. Научные интересы: управление сетевой инфраструктурой, операционные системы.

DOI: 10.15514/ISPRAS-2020-32(3)-7

An Approach to the Translation of Software-Defined Network Switch Flow Table into Network Processing Unit Assembly Language

A.A. Markoborodov, ORCID: 0000-0003-4525-6133 <amark@lvk.cs.msu.su>
Yu.A. Skobtsova, ORCID: 0000-0001-8351-3191 <xenerizes@lvk.cs.msu.su>
D.Yu. Volkanov, ORCID: 0000-0001-9940-5822 <volkanov@asvk.cs.msu.su>
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1

Abstract. This paper considers the OpenFlow 1.3 switch based on a programmable network processing unit (NPU). OpenFlow switch performs flow entry lookup in a flow table by the values of packet header fields to determine actions to apply to incoming packet (classification). In the considered NPU assembly language, lookup operation may be implemented on the basis of search trees. But these trees cannot be directly used for OpenFlow classification because of compared operands width limitation. In this paper, we propose flow table representation designed for easy translation into NPU search trees. Another goal was to create a compact program that fits in NPU memory. Another NPU limitation requires program updating after each flow table modification. Consequently, the switch must maintain the current flow table state to provide a fast NPU program update. We developed algorithms for incremental update of flow table representation (flow addition and removal). To evaluate the proposed flow table translation approach, a set of flow tables was translated into NPU assembly language using a simple algorithm (based on related work) and an improved algorithm (our proposal). Evaluation was performed on the NPU simulation model and showed that our approach effectively reduces program size.

Keywords: OpenFlow; network processing unit; flow table; software-defined network

For citation: Markoborodov A.A., Skobtsova Yu.A., Volkanov D.Yu. An Approach to the Translation of Software-Defined Network Switch Flow Table into Network Processing Unit Assembly Language. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 79-90. DOI: 10.15514/ISPRAS–2020–32(3)–7

Подход к трансляции таблицы потоков коммутатора программно-конфигурируемой сети в язык ассемблера сетевого процессора

А.А. Маркобородов, ORCID: 0000-0003-4525-6133 <amark@lvk.cs.msu.su>
Ю.А. Скобцова, ORCID: 0000-0001-8351-3191 <xenerizes@lvk.cs.msu.su>
Д.Ю. Волканов, ORCID: 0000-0001-9940-5822 <volkanov@asvk.cs.msu.su>
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1

Abstract. В статье рассматривается коммутатор, функционирующий под управлением протокола OpenFlow 1.3. Коммутатор работает на базе программируемого сетевого процессорного устройства (СПУ). Для классификации приходящих пакетов коммутатор выполняет поиск записи (правила) в таблице потоков по значениям полей заголовка для определения действий, которые необходимо выполнить над полученным пакетом. Поиск в программе на языке ассемблера рассматриваемого СПУ может быть реализован в виде набора деревьев поиска. При этом существует ограничение на ширину сравниваемых значений, что не позволяет напрямую использовать деревья поиска для

классификации по таблице потоков. В статье предлагается представление таблицы потоков, разработанное для трансляции таблицы потоков в программу на языке ассемблера СПУ, реализующую поиск по набору правил таблицы. Еще одной целью являлось создание компактной программы, которая может быть загружена в память СПУ. Архитектура рассматриваемого СПУ также обладает особенностью, заключающейся в необходимости обновления программы после каждого изменения таблицы потоков. Поэтому целесообразно поддерживать текущее представление таблицы потоков для быстрого обновления программы СПУ. В статье представлены алгориты для инкрементного обновления разработанного представления таблицы потоков (добавления и удаления правила). Разработанный подход был исследован на экспериментальных наборах правил, которые были транслированы в программы на языке ассемблера СПУ с использованием прямого способа, основанного на существующих подходах, и разработанного алгоритма. Экспериментальное исследование проводилось на основе модели СПУ и показало, что разработанный подход способен эффективно уменьшать размер программы.

Ключевые слова: OpenFlow; сетевое процессорное устройство; таблица потоков; программно-конфигурируемые сети

Для цитирования: Маркобородов А.А., Скобцова Ю.А., Волканов Д.Ю. Подход к трансляции таблицы потоков коммутатора программно-конфигурируемой сети в язык ассемблера сетевого процессора. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 79-90 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–7

1. Introduction

Software-Defined Networks (SDN) have been actively developed recently. In SDN network devices, or switches, implement data forwarding plane, when device and data flow management (control plane) is performed by special software – SDN controller, running on a separate server [1]. For interaction between the data plane and the control plane, a special control protocol is used. The OpenFlow [2] protocol is one of the most widespread SDN control protocols.

Packet processing in the OpenFlow switch is performed using special processing rules (called flow entries in the OpenFlow protocol) organized in flow tables. SDN controller updates these flow entries by sending OpenFlow messages. To classify incoming packets, OpenFlow switch looks up for the flow entry in the flow table that matches values of corresponding packet header fields.

One of the directions of the SDN technology development are high-performance switches based on programmable network processor units (NPU) [3], which are widely used. NPU is a System-on-a-Chip with architecture specialized for network traffic processing. NPU performs packet header parsing, classification of incoming packets, modification of the packet header and traffic management functions [4]. Programmable NPUs allow us to change packet processing algorithms and distinguishable packet header fields, which is highly valuable in SDN deployments with emerging standards like data centers or 5G [5].

NPU is a specialized device that executes packet processing program loaded into its memory and usually does not make changes to its program itself. The central processing unit (CPU) of the switch implements the interface with the SDN control plane. OpenFlow software in the operating system environment of the CPU provides a connection with the controller and makes changes to the NPU program. Program update requires a special system to translate OpenFlow abstractions into the assembly language of the NPU. This research is devoted to the development of such a system, specifically, its part responsible for packet classification according to the flow table.

The paper has the following structure. Section 2 describes the main architectural features of the NPU and its assembly language. Section 3 contains the problem statement of this research. In Section 4, we perform an analysis of related work applied to flow table representation in considered NPU. Section 5 presents developed data structure and algorithms for translating it into the assembly language and also the data structure updating algorithms. Section 6 is devoted to the evaluation of the developed flow table translation approach.

2. NPU architecture

Our research considers a switch with the NPU based on specialized computing cores. This NPU contains a set of parallel packet processing pipelines consisting of the uniform stages that execute binary code loaded into them.

The computing core of the NPU pipeline stage contains a single general-purpose register and a memory area to store the currently processed packet header and associated metadata (such as ingress port identifier). The register is used as an operand register and a result register. NPU does not contain a separated memory area to store program data. Program data is recorded directly to the binary code of the stage processor instructions. Thus, any change of the data, such as flow removal in OpenFlow, requires a new program to be loaded.

The assembly language of the NPU contains conditional jump instructions that compare the value in the register with the value from the instruction operand. Length of the value should be 64 bits or less. Conditional jump can be made only to the label located in the program below. Therefore, for example, it is impossible to implement loops or return to previously defined packet modifying action.

The program in the assembly language of the NPU can be represented as a finite set of linear instruction blocks connected by jump instructions. The program contains the following main types of linear instruction blocks.

- **Load value**. The sequence of instructions of this block loads a value from the packet header memory area or associated metadata memory into the register.
- **Change register**. The sequence of instructions of this block performs arithmetical or logical operations to change the current value of the register.
- **Search tree**. In the simplest case (exact match search tree) this block contains a set of jump instructions. The search key is an integer value, which length is 64 bits or less. It can also be the longest-prefix match search tree, which additionally requires prefix length for the searchkey.
- **Apply actions**. The sequence of instructions of this block performs modifying actions on the packet header memory or associated metadata memory, such as changing header field values, pushing tags.

A directed graph of the program can be created from the instruction blocks of the program. In this graph vertices are created for each instruction block. An arc leads from one vertex to another, if the block corresponding to the first vertex has jump instruction to label located in instruction block corresponding to the second vertex. The program graph has no cycles and contains only one vertex without incoming arcs.

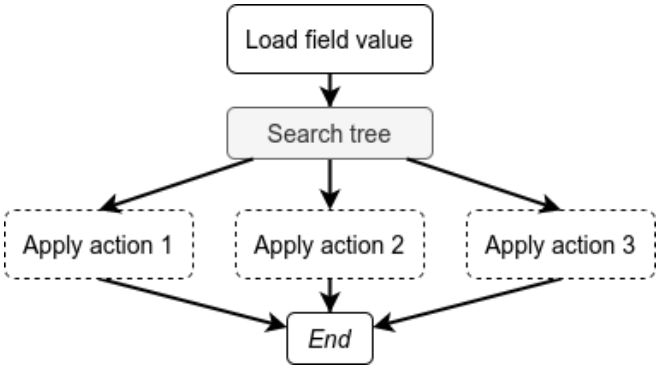


Fig. 1. Example NPU program graph

Fig. 1 shows the graph for a simple NPU program which performs classification by the value of one header field and applies one of the three packet modifying actions. The program has one instruction block for loading the value of this field and one block containing the search tree. In the program graph three arcs lead from the vertex of search tree block to three vertices of action blocks to apply.

3. Problem statement

Consider a switch that operates under the OpenFlow 1.3 protocol, based on the NPU described in Section 2. Let R be flow table with flow entries containing only match fields with exact values. The set of flow table match fields is denoted by $I = \{m_1; m_2; \dots ; m_k\}$. Flow entry may specify exact value only for a subset of I allowing any value in other match fields. Let the symbol «*» denote any value of the match field. To avoid search ambiguity, each flow entry is marked with priority p.

Our goal is to create a program in the assembly language that is compliant with the graph described in Section II and performs received packet classification by the given flow table R. The program must perform the search for matching flow entry in the flow table that has the highest priority and matches packet header fields.

Additionally, we have to load a new program into the NPU each time flow table contents are changed. Considering the usual frequency of flow table updates, it is advisable to maintain an incrementally updated intermediate representation of the flow table for quick translation after the update.

Thus, the problem is to develop a data structure for translating given flow table R into the program in the assembly language of the considered NPU, which implements a search on this set of flow entries and supports the addition and removal of flow entries.

4. Related work

This section provides a brief review of other researches devoted to data structures developed for classification by the flow table or similar multi-field tables.

The papers [6], [7] investigate an approach based on the decomposition of the classification by many fields into several classifications by one field. This approach uses a separate data structure for each match field, such as search trees or hash tables. The search result for one data structure is the Bloom filter [8] or label identifier. To get the classification result for all fields, it is necessary to intersect pairs of separate classification results. As a result, an identifier of the required flow entry is calculated.

This approach has significant limitations in implementation for the considered NPU, including the impossibility of hash function implementation required for Bloom filters and the necessity to store intermediate labels when classification results are intersected.

The papers [9], [10], [11], [12] suggest an approach that uses decision trees. Each vertex of such a tree is associated with a predicate. During the search, the predicate determines the next descendant vertex to continue the search. During passing from vertex to its descendant, the initial set of flow entries decreases and, as a result, turns into a smaller set of flow entries, among which the desired flow entry is determined by simple enumeration.

This approach also has limitations for implementing in the considered NPU. In the search process, it is required to load the header field values more than once, that can lead to unreasonable expenses for the packet processing time and multiple duplication of instruction blocks for loading the field value.

All considered approaches to the representation of flow tables have limitations and disadvantages for their implementation in the assembly language of considered NPU. Data structures based on decision trees are more suitable for our research problem. However, the disadvantages of such data structures should be eliminated, or the program just will not fit into NPU memory. 3

5. Proposed approach

In this section, we describe the developed data structure for representing the flow table and the developed algorithms for flow addition and removal from the data structure and for translating the data structure into a program in the assembly language of the NPU.

5.1 Data Structure

To represent a flow table with the set of flow entries R, we use a tree with marked vertices and arcs. The following values are associated with each tree vertex, except for leaf vertices.

- Match field from the set of considered fields $I = \{m_1; m_2; \ldots; m_k\}$: the tree root corresponds to the field m_1, the descendants of the root correspond to the field m_2, etc.
- Subset of the flow set *R*. The tree root corresponds to the whole set *R*.

Table 1. Example flow table

Flow	Priority	Field 1	Field 2
F_1	2	0	0
F_2	2	0	1
F_3	2	1	0
F_4	1	1	*

Each tree leaf is associated with a flow entry subset of *R* sorted in descending priority order. The tree has a depth of *k*, and all the tree leaves are vertices of the depth *k*. Table 1 presents an example flow table with two match fields, consisting of four flow entries F_1, F_2, F_3, F_4. In Fig. 2, the data structure constructed for example flow table that is shown in Table 1.

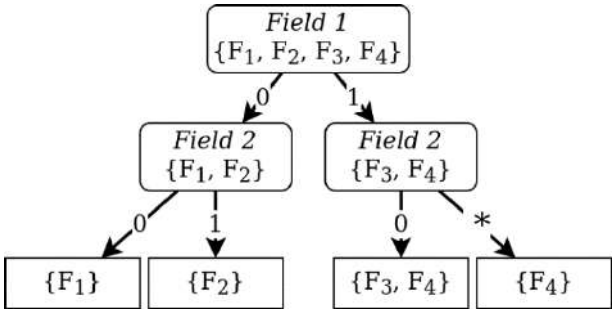


Fig. 2. Data structure constructed from flow entries given in Table 1

Consider the tree vertex *v*, which corresponds to the field *m* and a flow subset $S \subset R$. Then:

- if *M* is a set of all possible values of the field *m* in the flow entries from the set *S*, including the special value *, for each value $f \in M$ the vertex *v* has a descendant which arc is marked *f*;
- If the vertex *u* is a descendant of the vertex *v* with arc marked *f*, the flow subset of the vertex *u* contains only those flow entries from *S*, which value for the field *m* is *f* or *.

The developed data structure differs from approaches shown in related work by a fixed order of viewing fields. Vertices having the same depth refer to the same match field. This allows us to load the value of each match field only once in the search process. This order also allowed us to develop an algorithm for translating the data structure into the assembly language of the NPU, which receives the program without duplicating the instruction blocks for loading field value.

5.2 Flow Addition

Flow addition to the data structure is performed by traversing the tree vertices, starting from the root. When traversing a vertex, a new flow entry is added to its flow subset. Then, traversing

continues in vertex along the arc, which is marked with the value of the field specified in the added flow entry. If such an arc is in absent, a new vertex descendant is added.

When traversing a vertex, two special cases require additional actions.

1) The value of the field corresponding to the vertex specified in the added flow entry is *. In this case, in addition to the descendant along the arc marked *, it is necessary to traverse all other descendants of this vertex.
2) The value of the field corresponding to the vertex in the added flow entry is $f \neq *$ and this vertex has a descendant along the arc marked *. Then, in case of adding a new descendant, it is necessary firstly to copy the subtree corresponding to the arc marked * to the subtree along the arc marked *f*, and then continue the traversal.

5.3 Flow Removal

Flow removal from the data structure is also performed by traversing the tree vertices, starting from the root. When traversing a vertex, the removed flow entry is deleted from the vertex flow subset, and traversing continues in vertex along the arc, which is marked with the value of the field specified in the removed flow entry.

When traversing a vertex, two special cases require additional actions.

1) The value of the field corresponding to the vertex specified in the removed flow entry is *. In this case, in addition to the descendant along the arc marked *, it is necessary to traverse all the other descendants of this vertex.
2) The value of the field corresponding to the vertex in the removed flow entry is $f \neq *$, and this vertex has a descendant along the arc marked *. For this case, after traversing the subtree along the arc marked *f*, it is necessary to compare the subtree along the arc marked *f* and the subtree along the arc marked *. In case of equality, the subtree along the arc *f* is removed, because it is redundant.

5.4 Translation into NPU Assembly Language

Proposed data structure can be directly translated into a program in the assembly language of the NPU. The program graph will have a structure similar to a tree, but for each vertex of the tree, except for the leaves, the program graph will contain sequentially connected vertices corresponding to the instruction block of loading the field value, which corresponds to the vertex, and the instruction block of a search tree, for the values that mark the outgoing arcs from the vertex. The tree leaf will correspond to the instruction blocks of actions of the flow entry that has the highest priority in leaf flow set.

```
     <...> // Load Field 1
     tree in "tree_1"
     j End
L1:  // Load Field 2
     tree in "tree_2"
     j End
L2:  <...> // Load Field 2
     tree in "tree_3"
     j F4
F1:  <...> // Actions F1
     j End
F2:  <...> // Actions F2
     j End
F3:  <...> // Actions F3
     j End
F4:  <...> // Actions F4
     j End
```

```
End:
```

Listing 1. Program obtained by direct translation method

Listing 1 and Fig. 3 show the program and the program graph translated by the direct method from flow table representation presented in Fig. 2.

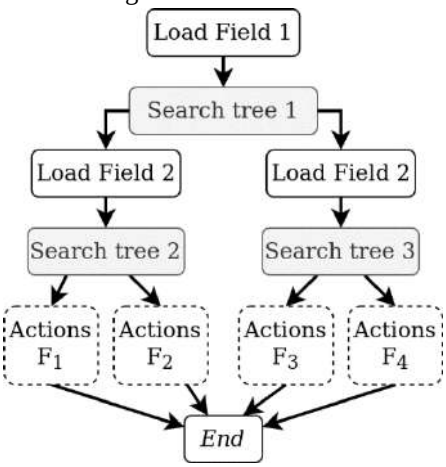


Fig. 3. Program graph obtained by direct translation method

However, with the direct method of translation, the resulting program contains a lot of search trees for similar key sets and duplicating instruction blocks for loading the field value (see duplicating field loading block in Fig. 3). To eliminate this drawback, a method for translating the data structure with encoding arcs was developed (method with encoding).

When translating by the method with encoding, tree levels are introduced. Tree level corresponds to the table match field and includes vertices of the same depth. The arcs outgoing from the tree levels vertices are numbered, that is, the code is assigned to each arc. Numbering for each level is independent. Then, for each tree level, a level list, consisting of all pairs (code of the incoming arc, marker of the outgoing arc) is formed. For the root vertex, zero is used instead of the incoming arc code. Fig. 4 shows arc encodings, tree levels, and level lists for our example data structure.

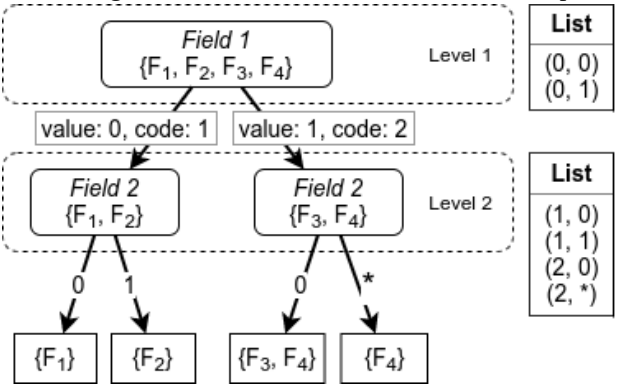


Fig. 4. Flow table representation marked up with tree levels encoded arcs and level lists

Instruction blocks of the search tree and loading field corresponding to the level are created for each list of pairs. Jumps in the block of the search tree are performed in a special block for loading the code of the outgoing arc, to which the pair corresponds. Then jump is performed to loading the value of the next field. In Fig. 5, the resulting program graph, obtained by the method with encoding from marked flow table representation in Fig. 4, is shown. In Listing 2, the corresponding program in the assembly language is presented.

Thus, the resulting program contains one loading instruction block for each match field and a fixed number of search trees, one search tree per match field.

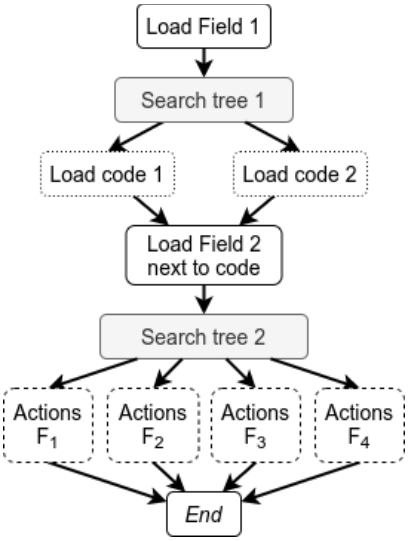


Fig. 5. Program graph obtained by translation method with encoding

Listing 2. Program obtained by translation method with encoding

```
      <...> // Load Field 1
      tree lpm "tree_1"
      j End
L1:   loadi 1
      j L3
L2:   loadi 2
      j L3
L3:   rol FIELD_2_WIDTH
      <...> // Load Field 2
      tree lpm "tree_2"
      j End
F1:   <...> // Actions F1
      j End
F2:   <...> // Actions F2
      j End
F3:   <...> // Actions F3
      j End
F4:   <...> // Actions F4
      j End
End:
```

6. Evaluation

For the developed data structure, we evaluated translation method with encoding in comparison to the direct translation method inspired by approaches from related work.

For the evaluation, we used a simulation model of the NPU pipeline. The simulation model receives a program in the assembly language, translated by one of the methods in our case, and a set of input packets to be processed. As an output the model produces a set of outgoing packets along with statistics, including the amount of memory occupied by the program binary code and the average number of ticks spent on processing the input packets. Before processing the packets, the simulation model translates the program in assembly language into binary code, where each instruction has 16 bytes length.

We generated a set of OpenFlow tables with match fields of the data link (L2) and network layer (L3) header fields with different numbers of flow entries. For each table, the evaluated data structure was built and translated into NPU assembly language. For each flow table, one input packet per flow entry was generated.

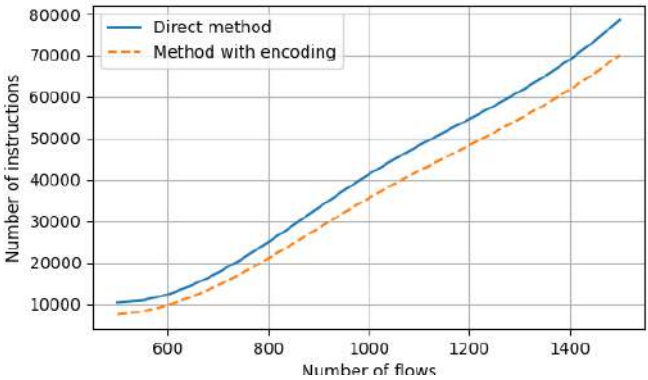


Fig. 6. Average number of instructions for different flow table sizes

The dependency between the number of table flow entries and the average number of instructions is presented in Fig. 6. The measurements show that the program translated by the direct method takes from 1.2 to 1.5 times more memory than the program translated by the method with encoding.

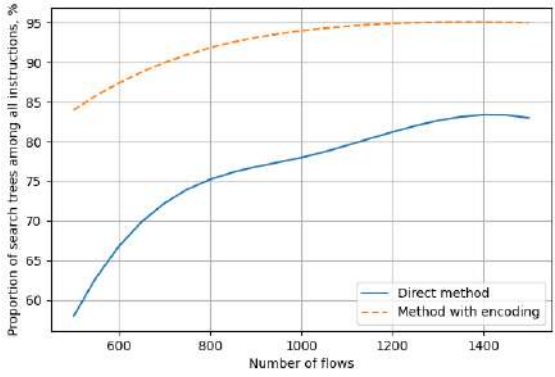


Fig. 7. Proportion of the search trees instructions for different flow table sizes

Fig. 7 shows the proportion of the search trees instructions in the program depending on the number of table flow entries. The proposed method with encoding uses memory much more effectively than the direct method, removing from 15 to 30% duplicating code from the program.

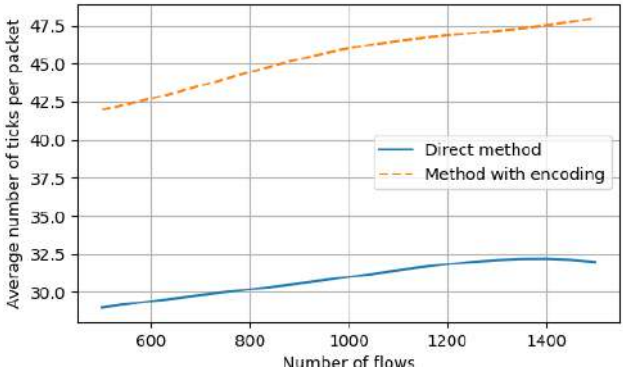


Fig. 8. Average number of ticks per packet for different flow table sizes

However, the fee for reducing the size of the program is an increase in the number of ticks per packet, which is about 10%. In Fig. 8, the dependency between the number of flow entries in the table, and the average number of ticks per packet is rendered. The increase in packet processing time, though, is still within acceptable limits for our NPU and does not lead to unexpected delays or packet drops.

In future research, we are going to determine the dependencies of the evaluated characteristics by the number of match fields in the flow entries. All of the proposed algorithms, including flow addition and removal, will be evaluated in terms of data structure update time.

7. Conclusion

In our research, we considered the switch based on programmable NPU, which has architectural limitations in memory organization. To use this NPU in the SDN switch operating under the OpenFlow 1.3 protocol, the system for translating flow table into NPU program was developed. The system allows us to get a program for the NPU with acceptable packet processing time, which takes up to 30% less memory comparing to the programs based on data structures in the considered related work. These results are achieved by reducing the duplication of instruction blocks that load the value of the same fields, reducing the program size (in some cases by 1.5 times). In the future, we will consider maskable match fields of the flow entries and examine the effect of the fields parsing order on the resulting program characteristics.

References / Список литературы

[1]. Open Networking Foundation. Software-defined networking: the new norm for networks. ONF white paper, 2012. Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[2]. Open Networking Foundation. OpenFlow switch specification version 1.3.0. 2012. Available at: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf.

[3]. Giladi R. Network processors: architecture, programming, and implementation. Morgan Kaufmann, 2008, 736 p.

[4]. Orphanoudakis T., Perissakis S. Embedded multi-core processing for networking. In Embedded Multi-Core Systems, CRC Press, 2010, pp. 399–463.

[5]. Bifulco R., Rtvri G. A survey on the programmable data plane: abstractions, architectures, and open problems. In Proc. of the IEEE 19th International Conference on High Performance Switching and Routing (HPSR), 2018, pp. 1–7.

[6]. Taylor D., Turner J. Scalable packet classification using distributed crossproducting of field labels. In Proc. of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 2005, pp. 269-280.

[7]. Kekely M., Korenek J. Packet classification with limited memory resources. In Proc. of the Euromicro Conference on Digital System Design (DSD), 2017, pp. 179–183.

[8]. Bloom B. H. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, vol. 13, no. 7, 1970, pp. 422–426.

[9]. Gupta P., McKeown N. Classifying packets with hierarchical intelligent cuttings. IEEE Micro, 2000, vol. 20, no. 1, pp. 34–41.

[10]. Singh S., Baboescu F., Varghese G., Wang J. Packet classification using multidimensional cutting. In Proc. of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2003, pp. 213–224.

[11]. Qi X., Xu L., Yang B. Packet classification algorithms: from theory to practice. In Proc. of the IEEE International Conference on Computer Communications (IEEE INFOCOM), 2009, pp. 648–656.

[12]. Li W., Li X., Li H., Xie G. CutSplit: a decision-tree combining cutting and splitting for scalable packet classification. In Proc. of the IEEE International Conference on Computer Communications (IEEE INFOCOM), 2018, pp. 2645–2653.

Information about authors / Информация об авторах

Andrei Aleksandrovich MARKOBORODOV – student of the faculty of the CMC. Research interests: software-configured networks, network processor units.

Андрей Александрович МАРКОБОРОДОВ – студент факультета ВМК. Научные интересы: программно-конфигурируемые сети, сетевые процессорные устройства.

Julia Alexandrovna SKOBTSOVA – specialist, faculty of the CMS, department of automation of computer systems, laboratory of computer systems. Research interests: software-configurable networks, network processor units, hardware description languages.

Юлия Александровна СКОБЦОВА – специалист, факультет ВМК, кафедра автоматизации систем вычислительных комплексов, лаборатория вычислительных комплексов. Научные интересы: программно-конфигурируемые сети, сетевые процессорные устройства, языки описания аппаратуры.

Dmitry Yuryevitch VOLKANOV – candidate of physical and mathematical sciences, associate professor. Areas of research: analysis and design of network processing unit architecture.

Дмитрий Юрьевич ВОЛКАНОВ – кандидат физико-математических наук, доцент. Направления исследований: анализ и разработка архитектуры сетевого процессора.

DOI: 10.15514/ISPRAS-2020-32(3)-8

Analysis of student activity on the e-learning course based on «OpenEdu» platform logs

N.D. Barsukov, ORCID: 0000-0003-3962-9087 <nik0xff@gmail.com>
I.M. Sysoev, ORCID: 0000-0001-5748-5529 <ivanabc97@gmail.com>
A.A. Pereskokova, ORCID: 0000-0002-6937-150X <alina.alexandrovna.sh@gmail.com>
I.V. Nikiforov, ORCID: 0000-0003-0198-1886 <igor.nikiforovv@gmail.com>
D. Posmetnijs, ORCID: 0000-0001-9573-9286 <posmetnijs@gmail.com>

Peter the Great St.Petersburg Polytechnic University,
29, Polytechnicheskaya, St.Petersburg, 195251, Russia

Abstract. A lot of people nowadays use online education platforms. Most of them run on the free «Open edX» open-source software platform. Using the logs that the platform provides us, we can get psychometrics of students, which can be used to improve the presentation of material or other things, which can increase the quality of online courses. We provide a ready-to-use tool that will help figure out how and for what purpose you can analyze the log files of platforms based on «Open edX».

Keywords: Big Data; e-learning; analytics; logs; online educational platforms; microservices

For citation: Barsukov N.D., Sysoev I.M., Pereskokova A.A., Nikiforov I.V., Posmetnijs D. Analysis of student activity on the e-learning course based on «OpenEdu» platform logs. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 91-100. DOI: 10.15514/ISPRAS–2020–32(3)–8

Анализ активности студентов на курсах онлайн-обучения на основе логов платформы «OpenEdu»

Н.Д. Барсуков, ORCID: 0000-0003-3962-9087 <nik0xff@gmail.com>
И.М. Сысоев, ORCID: 0000-0001-5748-5529 <ivanabc97@gmail.com>
А.А. Перескокова, ORCID: 0000-0002-6937-150X <alina.alexandrovna.sh@gmail.com>
И.В. Никифоров, ORCID: 0000-0003-0198-1886 <igor.nikiforovv@gmail.com>
Д. Посметныйс, ORCID: 0000-0001-9573-9286 <posmetnijs@gmail.com>

Санкт-Петербургский политехнический университет Петра Великого,
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29

Abstract. В настоящее время многие люди используют образовательные онлайн-платформы. Большинство из них работают на бесплатной программной платформе с открытым исходным кодом «Open edX». Используя логи, которые предоставляет нам платформа, мы можем получить психометрические данные студентов, которые можно использовать для улучшения представления материала или других вещей, которые могут повысить качество онлайн-курсов. Мы предоставляем готовый инструмент, который поможет выяснить, как и с какой целью вы можете анализировать лог-файлы на основе «Open edX».

Ключевые слова: большие данные; онлайн-обучение; аналитика; логи; платформы для онлайн-обучения; микросервисная архитектура

Для цитирования: Барсуков Н.Д., Сысоев И.М., Перескокова А.А., Никифоров И.В., Посметныйс Д. Анализ активности студентов на курсах онлайн-обучения на основе логов платформы «OpenEdu». Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 91-100 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–8

1. Introduction

Online electronic educational learning platforms are very popular nowadays. One of the biggest and widely used platforms is «Open edX» [1-3]. This is an open-source software platform that provides off-the-shelf tools for educational services. One of the important features of the platform is that it generates student and teacher activity log files. But the disadvantage of the platform is that it does not provide any data analysis tools for monitoring educational progress and success.

One of the most popular educational platform based on «Open edX» in our region is «Open Education» [4]. The problem of the Open edX platform and in particular of one of its implementations «Open Education» is that the teachers, conducting courses on this platform, are missing the tools for analyzing the educational process, which leads to missing control on the educational process and decreasing its efficiency. On the other hand, the students, who use the platform, are also not able to monitor their academic performance.

It's important to provide teachers, course administrators, and students with educational analytics tools that help them to make the educational process more efficient [5] Improving online educational platforms can make online learning at universities more friendly, easy, and happy for all the involved actors.

Our work and project aim consist of several important parts:

- make research on the structure and format of the Open edX platform logs for applicability for automatic analysis of the students' performance. The result of the research showed that all required and important user activity actions (audits) are presented in the logs, so the analytics is possible. Also, we realized that the size of the logs is huge, and they contain millions of actions logged, which makes us think about the usage of Big Data technologies for analytical purposes;
- create and formulate analytic tasks, that can be solved on the logs. The result of that activity is that there is a list of 18 analytics tasks that help students and teacher to monitor the progress;
- implement the software solution, that demonstrates the idea and all the possibilities that Open edX logs provide. As an outcome, there is a tool for extracting, transforming, and preserving logs from a specific course / courses from this platform and the number of analytics tasks implemented in that tool. The result of the analysis is presented in the form of files with metrics, and graphs based on the data obtained.

The paper has the following structure. Section 2 shows the related work. Section 3 describes the system design. Section 4 discusses the implementation process and result of the pilot project. Section 5 presents the conclusion.

2. Related work

There are some articles facing the similar problem of online courses activity analysis.

In [6] (below, Article 1), authors take Moodle as a target platform of further analysis. Authors also use log files as a data source for further analysis. The files are stored in a database, processed and visualized to provide data implementation for teachers who are the final users of that tool. The metrics which authors take for analysis are "the grades of online assignments, reading time, the total number of login times, the total number of online discussions" and others. The choice of these metrics is based on log files content – such values may be easily extracted from raw data.

Speaking about analytics, authors of [7] (Article 2) use advanced machine learning methods such as Random Forest to provide analytics of the online learning process. Particularly, the work describes the prediction of a student's dropout from a course.

Authors of [8] (Article 3) also suggest a method for detection of students who seem to be expelled at the end of the course. Authors use machine learning methods to make the prediction of the further academic performance of students. The prediction is based on logged data of the educational platform.

In [9] (Article 4), authors make statistical analysis of the online course data. The course is running on Moodle platform. Authors also use a self-made logging system to extend log data provided by the platform with new types of recorded events.

Authors of [10] (Article 5) make the visualization of LMS log data. Firstly, they make preprocessing of logs and then draw the scatter plot of student activity within a specific class and the plot of whole faculty activity during one online course.

We've made an analysis of these papers using 5 characteristics. These characteristics describe each of the solutions proposed by papers' authors. The characteristics are:

- *name of educational platform used for analysis* – the name of online educational platform used in article;
- *log files were used for analysis* – did authors use log files to make analysis or not;
- *prediction methods were used in analysis* – do authors use prediction methods in their analysis or not;
- *data visualization was made* – did authors make a visual interpretation of their results or not;
- *eeady-to-use tool was developed* – have authors developed a ready-to-use tool for third-party usage or not.

The result of related works analysis is presented in Table 1.

Table I. Articles' comparison analysis

	Paper 1	***Paper 2***	***Paper 3***	***Paper 4***	***Paper 5***
Name of educational platform used for analysis	Moodle	Not mentioned	Not mentioned	Moodle	Moodle
Log files were used for analysis	+	—	+	+	+
Prediction methods were used in analysis	—	+	+	+	—
Data visualisation was made	+	+	+	+	+
Ready-to-use tool was developed	+	—	—	—	—
Legend: "+" - supported; "—" - unsupported					

According to the result of related works analysis, all of the authors make analysis of student activity during one or more online courses and all of them make the visual interpretation of the results. Most authors base their results on log files data from educational platforms. Talking about platforms, Moodle is the most popular one within paper authors. Some papers also contain descriptions of prediction methods to make forecasts of student academic performance.

However, only one paper describes a ready tool which contains all analytics methods described there and which can be utilized by other people. Also, any of these articles doesn't describe work with the Open edX platform.

Within our work we are going to implement a tool for analysis of online courses data based on log files of the Open edX platform.

Ultimately, our tool differs from others in that we directly process the platform logs, which allows flexibility in the approach to the analysis of what happened on the course. Thus, the teacher can get an answer to the question on a very specific task, in contrast to other tools.

3. System design

In our solution, we are using microservice architecture presented in fig. 1. This allows us to conveniently implement and modify the logic of the service rebuild and redeploy a small part of the tool instead of full application rebuilding. We are using Docker and other DevOps practices. Docker effectively helps us in leveraging microservices architecture [11]. We see three microservices here.

- UI Service – allows the end-user to interact with the application.
- ETL Service – is responsible for receiving, transforming unstructured logs, and loading them into the Database; This service can receive logs from the local machine or directly from the platform based on «opened». In our case, this is the platform – «Open education».
- The analytical service contains a set of analytical scripts that work with the database and a module for building the output file - result or report, which will be sent to the appropriate user interface.

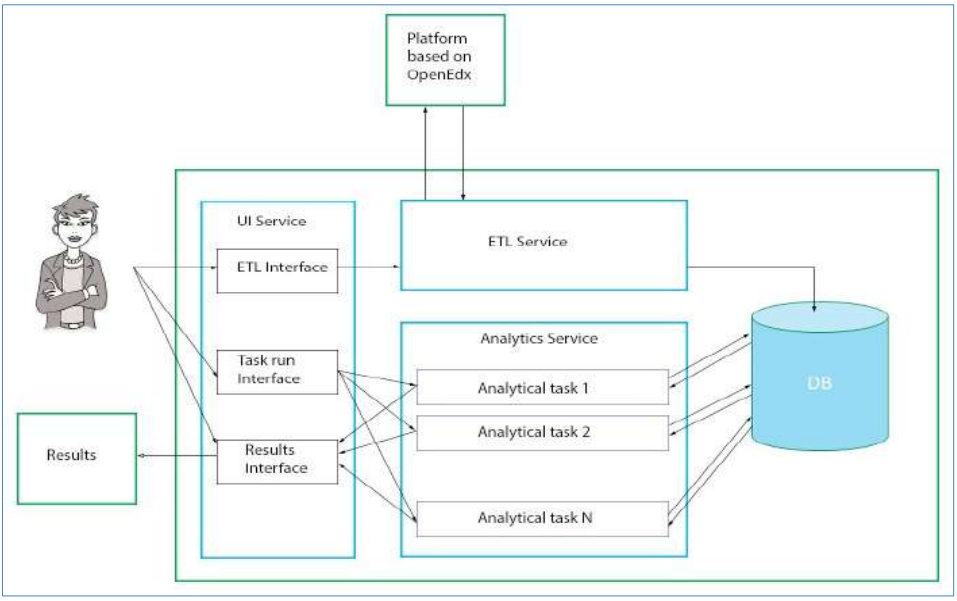


Fig. 1. Architecture of the application

3.1 Log-file structure

All educational platforms can maintain the activity (actions) of users on the platform while undergoing learning on the course or performing test and examination tasks By this way, it becomes possible to analyze user behavior and, based on the obtained analytics, improve educational courses and receive psychometric [12-13] data of students. Due to the improvement of the courses, it will be more convenient and easier for students to learn the material obtained, it will be easier for teachers to distinguish distinguished students and more accurately set final grades.

The typical log file is presented in fig. 2. It is a JSON file describing the events occurring in the LMS system [14]. An example of a log-file is shown in figure 1. An event is an entity that describes individual user activity in a course (for example, enrolling in a course, watching a video lecture, sending a response during testing, etc.). In the log file, the event is represented by a JSON object and contains a set of fields.


```
{
  "ip":"anon_22e8ad25256ffdc60a8cb8957e4ebdb7dc8f3f62a2eac77a285b1bdb1c0d82ea",
  "host":"courses.openedu.ru",
  "page":null,
  "time":"2018-09-09T06:35:17.172512+00:00",
  "agent":"Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
  "event":"{\"POST\": {}, \"GET\": {}}",
  "context":{
    "path":"/api/extended/calendar/course-v1:spbstu+PHYLOS+fall_2018",
    "org_id":"",
    "user_id":955006,
    "course_id":""
  },
  "referer":"https://openedu.ru/my/",
  "username":"alive-proud-snarespenguin",
  "event_type":"/api/extended/calendar/course-v1:spbstu+PHYLOS+fall_2018",
  "event_source":"server",
  "accept_language":"ru,en;q=0.9"
}
```


Fig. 2. Log example

Among the fields that are most interesting as part of the analysis tasks, one can single out the time field (the time the event was recorded in the log file), user_id (user identifier - the initiator of the event), course_id (course identifier) and event_type (the type of event listed in the documentation). A complete list of events used in Open edX is given in the documentation.

3.2 User interfaces

The latest stable version of our tool provides the CLI (command-line interface). After starting the tool, the user is asked about the logs that he would like to analyze, and all available log names are shown to the user in a list. The user must select the name of the log and enter it. The next step is to select an analysis task. The user will see all available tasks and will have to enter the number of the selected task for analysis.

Some tasks require additional input parameters to run. If the user selects such a task, he will be given the corresponding messages in the console, and the user will have to enter the necessary parameters, such as launching the task for all users or only one selected user. After starting this task, the user must wait for its completion. The user will receive a message with information about the placement of the results, and if any graphs are created after the task is completed, they will be automatically opened in the browser.

In addition, we are developing a new graphical user interface that is not yet included in the stable version. It consists of several pages on which the user can see the instructions, select the logs, start the analysis, and see the results. The graphical user interface is more user-friendly.

We have our log analysis algorithm. Using a query in the database, we get the information necessary for analysis from the log, then we analyze it using various mathematical techniques. In one of the tasks, we use some innovation.

3.3 Innovative aspects of the design

Since the tasks of analytics depend on the requirements of the customer, it is necessary to create such a backend architecture in which we can easily integrate new tasks for analytics. OOP architecture and Reflections API is well suited for solving this problem, so we can add new tasks inherited from an abstract task by adding only the database queries without changing the architecture of the project.
At each startup, the system overloads the log, which makes it resistant to user crashes and software error implementation.

4. System implementation

4.1 Description of the implementation

We use laptops for development and production deployment. Our tool is designed for teachers who create their courses and for administrators of educational platforms based on the Open Source "Open edX" software platform. We designed our tool for any hardware platform which meets minimum requirements: 4GB RAM, 2.5Ghz processor.
For user interface we use React, Redux libraries, webpack, and Babel for modules bundling and converting JavaScript code to backward-compatible representation. Npm is a package manager we use for handling packages in development. Lodash is a modern JavaScript utility library delivering modularity, performance, and extras. (See Table 2 for licensing information.)
For analytical and ETL services we are using Java 8 with Spring 5 [15]. For better readability of Java code, we use the Lombok library [16], which allows you to reduce the boilerplate code (constructors, «Object»-methods and etc.) by using annotations. Reflection allows us to look at existing tasks in the project, create a list of tasks and give it to the UI. For documentation we use swagger, it allows us to create documentation in a semi-automatic mode for our services.
We use PostgreSQL [17] as a database. Our build system – Gradle [18-19]. Interaction between database and service is provided by JDBC [20] driver. Our system is RESTful. All microservices use the REST paradigm to interact with each other.

Table 2. Components acquired from external sources

Library	**License**
Spring	Apache License 2.0
Springfox-swagger	Apache License 2.0
Reflections	BSD 2-clause
Lombok	MIT License
React	MIT License
React-bootstrap	MIT License
Redux	MIT License
Webpack	MIT License
Lodash	MIT License

4.2 Innovative aspects of the implementation

In the implementation of our system, we decided to record each log file (which is a JSON-file) as every row in our PostgreSQL database. So, we use it like a No-SQL database. SQL query for logs selection is presented in fig. 3.
PostgreSQL has many standard functions such as working with strings or JSON-files, which allow working with logs more effectively. We are thinking about transferring our database to No-SQL [21] in the future if it is proved that it will be more productive.

```
SELECT log_line -> 'username' as user_name,
       log_line #>> '{context, user_id}' AS user_id
FROM logs
WHERE log_line -> 'username' != 'null'
```

Fig. 3. Select query example

5. Results

5.1 System output

As a result, we received a solution that provides course administrators the following functionalities:

- download log files from educational platforms;
- choose log files between different courses on a platform;
- store log files in a database;
- run analytic tasks based on downloaded logs, a few of them:
  1) Calculate total user time on the course and user time distributed per day;
  2) Show activity type for all users (or for a particular user) on course depending on the date;
  3) Show the user way over the pages;
  4) Show amount of video play events per day;
  5) Show words from pdf search field;
  6) Get video watching durations by elements of course.

The system gives the calculations in the form of tables saved in XLSX or CSV format and generated graph from these tables.

On the graph in figure 4 we can see the rule that the user visited on the course. We can conclude that this user watches the lectures for two months. After that he decided not to continue studying but we can write him a letter and find his opinion perhaps he didn't like this course or problem was in the poor presentation of material.

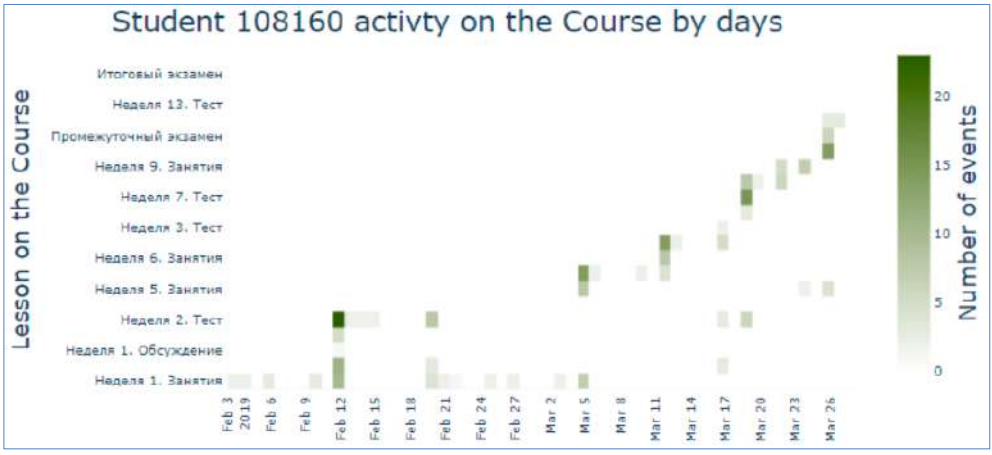


Fig. 4. User activity on course graph

5.2. Challenge and issues

The main challenge was to study the logs and their loading, conversion, and analysis. It was required to read and learn a lot of Open edX documentation. On the other hand, the initial log file that we took was 18Gb to analyze, which made it almost impossible to leant it manually, so we had to create

simple pasting scripts to learn it fast, and only then we were able to retrieve a small log for testing purposes.

Our current database design has limitations that do not allow us to process the logs as fast as we wanted. Therefore, we have found a solution that is now improving. Logs can be larger than 18 GB and contain more than 10 000 000 events generated with semi-structured information inside, so the problem is to process the logs quickly enough.

Open edX platform doesn't provide a good and quick API for logs downloading for offline analysis, so that feature request and feedback has been provided to the platform administrators, but unfortunately, until today they have not implemented the required functionality for us.

References / Список литературы

[1]. Open edX official website, Edx Documentation Resources. Available at: https://docs.edx.org/

[2]. Blagojević, M., Milošević D.: Massive open online courses: EdX vs Moodle MOOC. In Proc. of the 5th International Conference on Information Society and Technology, 2016, pp. 346-351.

[3]. Sriram M. Comparative Analysis of Massive Open Online Course (MOOC) Platforms. In Proc. of the 4th International Conference on Global Business, Economics, Finance and Social Sciences, 2015, pp. 1-7.

[4]. Open Education official website, About the Project. Available at: http://npoed.ru/about (in Russian).

[5]. Krasnov S., Kalmykova S., Abushova E., Krasnov A. Problems of Quality of Education in the Implementation of Online Courses in the Educational Process. In Proc. of the International Conference on High Technology for Sustainable Development (HiTech), 2018, pp. 1-4.

[6]. M. Furukawa, K. Yamaji, Y. Yaginuma and T. Yamada. Development of learning analytics platform for OUJ online courses. In Proc. of the IEEE 6th Global Conference on Consumer Electronics (GCCE), 2017, pp. 1-2.

[7]. B.B. Mishra and S. Mishra. Quality Improvements in Online Education System by Using Data Mining Techniques. In Proc. of the 2nd International Conference on Data Science and Business Analytics (ICDSBA), 2018, pp. 532-536.

[8]. Nobuhiko Kondo, Midori Okubo, Toshiharu Hatanaka. Early Detection of At-Risk Students Using Machine Learning Based on LMS Log Data. In Proc. of the 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 2018, pp. 198-201.

[9]. P. Esztelecki . G. Korosi. Analysis of a short on-line course through logged data recording by a self-developed logging module. In Proc. of the International Conference on Computer, Information and Telecommunication Systems (CITS), 2018, pp. 1-5.

[10]. R. Raga, Jennifer Raga. A comparison of college faculty and student class activity in an online learning environment using course log data. In Proc. of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, 2017, pp. 1-5.

[11]. Jaramillo D., Nguyen D., Smart R. Leveraging microservices architecture by using Docker technology 2016. In Proc. of the IEEE Region 3 South East Conference (SoutheastCon), 2016, pp. 1-5.

[12]. Kalmykova S.V., Chapaykina M.D., Shirokova S.V. Application of a systematic approach to the project management implementation of e-learning in a classical university (on the example of the Peter the Great University). Obrazovatel′nye tehnologii, no. 2, 2018, pp. 67-74 (in Russian) / Калмыкова С.В., Чапайкина М.Д., Широкова С.В. Применение системного подхода к управлению проектом внедрения электронного обучения в классическом университете (на примере ФГАО ВО СПбПУ Петра Великого). Образовательные технологии, no. 2, 2018 г., стр. 67-74,

[13]. Kravchenko D. Psychometrics in online education. University Book: Journal of Information and Analysis, no. 3, 2019, pp. 52-55 (in Russian) / Кравченко Д. Психометрика в онлайн-образовании. Университетская книга: информационно-аналитический журнал, no. 3, 2019 г., стр. 52–55.

[14]. Edx, Student Events. Available at: https://edx.readthedocs.io/projects/devdata/en/latest/internal_data_formats/tracking_logs/student_event_types.html

[15]. Spring official website, Microservices. Available at: https://spring.io/microservices.

[16]. Project Lombok Official website, Lombok Features. Available at: https://projectlombok.org/features/all.

[17]. PostgreSQL official website, Documentation. Available at: https://www.postgresql.org/docs/10/index.html

[18]. Gradle official website. Gradle Features. Available at: https://gradle.org/features/.

[19]. Voinov N., Rodriguez Garzon K., Nikiforov I., Drobintsev P. Big Data Processing System for Analysis of GitHub Events. In Proc. of the 2019 XXII International Conference on Soft Computing and Measurements (SCM), 2019, pp. 187-190

[20]. Oracle Help Center, Lesson: JDBC Introduction. Available at: https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html.

[21]. Strauch Christof. NoSQL Databases. 2012. Available at: https://www.christof-strauch.de/nosqldbs.pdf.

Information about authors / Информация об авторах

Nikita Dmitrievich BARSUKOV is a graduate student at the Institute of Computer Science and Technology. Research interests: development of highly loaded software, development of corporate software systems, big data analytics, financial technologies, software verification.

Никита Дмитриевич БАРСУКОВ – студент магистратуры Института компьютерных наук и технологий. Область интересов: разработка высоконагруженного программного обеспечения, разработка корпоративных программных систем, аналитика больших данных, финансовые технологии, верификация программного обеспечения.

Ivan Mikhailovich SYSOYEV – graduate student at the Institute of Computer Science and Technology. Research interests: Big Data, Big Data Analytics, Data Model Building, Data Visualization.

Иван Михайлович СЫСОЕВ – студент магистратуры Института компьютерных наук и технологий. Область интересов: большие данные, аналитика больших данных, построение моделей данных, визуализация данных.

Alina Aleksandrovna PERESKOKOVA – graduate student at the Institute of Computer Science and Technology. Area of interest: big data, big data analytics, distributed data storage systems, interface development, big data display.

Алина Александровна ПЕРЕСКОКОВА – студентка магистратуры Института компьютерных наук и технологий. Область интересов: большие данные, аналитика больших данных, системы распределенного хранения больших данных, разработка интерфейсов, отображение больших данных.

Igor Valerievich NIKIFOROV – candidate of technical sciences, associate professor of the Higher School of Software Engineering at the Institute of Computer Science and Technology. Research interests: parallel data processing, distributed data storage systems, big data, software verification, test automation.

Игорь Валерьевич НИКИФОРОВ – кандидат технических наук, доцент Высшей школы программной инженерии Института компьютерных наук и технологий. Сферы научных интересов: параллельная обработка данных, системы распределенного хранения данных, большие данные, верификация программного обеспечения, автоматизация тестирования.

Deniss POSMETNIJS – graduate student at the Institute of Computer Science and Technology. Research interests: Big Data, Big Data Analytics, Development of End User Interfaces (Front End), Data Visualization.

Денисс ПОСМЕТНЫЙС – студент магистратуры Института компьютерных наук и технологий. Область интересов: большие данные, аналитика больших данных, разработка интерфейсов конечного пользователя (фронтенд), визуализация данных.

DOI: 10.15514/ISPRAS-2020-32(3)-9

Recommendation system based on user actions in the social network

V.V. Monastyrev, ORCID: 0000-0001-6770-4481 <vit34-95@mail.ru>
P.D. Drobintsev, ORCID: 0000-0003-1116-7765 <drobintsev_pd@spbstu.ru>
Peter the Great St.Petersburg Polytechnic University,
29, Polytechnicheskaya, St.Petersburg, 195251, Russia

Abstract. Currently, a large number of people use various photo hosting services, social networks, online services, and so on. At the same time, users leave a lot of information about themselves on the Internet. These can be photos, comments, geotags, and so on. This information can be used to create a system that can identify different target groups of users. In the future, you can run ad campaigns based on target groups, create recommendation ads, and so on. This article will discuss a system that allows users to identify their interests based on their actions in a social network. The following features were selected for analysis: published photos and text, comments on posts, information about favorite publications, and geotags. To identify target groups, the task was to analyze images in photos and analyze text. Image analysis involves object recognition, and text analysis involves highlighting the main theme of the text and analyzing the tone of the text. The analysis data is combined using a unique identifier with the rest of the information and allows you create a data showcase that can be used to select target groups using a simple SQL-query.

Keywords: machine learning; recommendation system; natural language processing; image recognition

For citation: Monastyrev V.V., Drobintsev P.D. Recommendation system based on user actions in the social network. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 101-108. DOI: 10.15514/ISPRAS–2020–32(3)–9

Рекомендательная система на основе действий пользователей в социальной сети

В.В. Монастырев, ORCID: 0000-0001-6770-4481 <vit34-95@mail.ru>
П.Д. Дробинцев, ORCID: 0000-0003-1116-7765 <drobintsev_pd@spbstu.ru>
Санкт-Петербургский политехнический университет Петра Великого,
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29

Abstract. В настоящее время большое количество людей пользуются различными социальными сетями, онлайн-сервисами и тому подобное. При этом пользователи оставляют различную информацию в подобных системах. Это могут быть фотографии, комментарии, геотеги и так далее. Эта информация может быть использована для создания системы, которая может идентифицировать различные целевые группы пользователей. На основе этой информации можно запускать рекламные кампании, создавать рекомендательные объявления и много другое. В данной статье рассматривается система, которая позволяет идентифицировать интересы пользователей на основе их действий в социальной сети. Для анализа были выбраны следующие типы данных: опубликованные фотографии и текст, комментарии к записям, информация о любимых публикациях и геотеги. Для выявления целевых групп была поставлена задача проанализировать изображения на фотографиях и проанализировать текст. Анализ изображений включает в себя распознавание объектов, а анализ текста включает в себя выделение основной темы текста и анализ тональности текста. Данные анализа объединяются с помощью уникального идентификатора с остальной информацией и позволяют создать витрину данных, которая может быть использована для поиска целевых групп с помощью простого SQL-запроса.

Ключевые слова: машинное обучение; рекомендательная система; обработка естественного языка; распознавание изображений

Для цитирования: Монастырев В.В., Дробинцев П.Д. Рекомендательная система на основе действий пользователей в социальной сети. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 101-108 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–9

1. Introduction

Currently, humanity actively uses various Internet services and leaves a lot of different data on the Internet. This can be photos, text information, and so on. Based on this information, you can divide users into groups based on their interests. Many companies have their own recommendation systems that operate on this principle – Yandex [1], Google (YouTube) [2], Netflix [3].

In this article, we will look at a recommedation system that will identify interest groups based on the following data: photos, text, rated publications, and geotags. The final goal is to create a target data table (in SQL format). From the SQL table, you can get a list of users based on the specified interest using an SQL query. To create such a table, you need to recognize objects in images, and recognize the main theme and tone in the text. This will help you understand which topics the user treats positively, which ones negatively, and which ones are neutral.

Thus, the final table will contain information about what the user posts, what they comment on, what and how they evaluate, as well as information about geolocation. Based on this information, which is specific to a particular user, you can easily get different groups of users by interests and geolocation.

2. Existing recommendation systems

As mentioned above, many large companies use different recommendation systems to process their data. It all depends on the specific task and the available data, so companies build the data processing process in a way that is convenient for them and usually such solutions are not open source. These can be systems for recommending movies, music, friends, interesting authors, and so on. Let's look at some of them in more detail.

To generate a smart news feed, the social network Vkontakte marks data with the help of users who have received the status of experts [4]. These users vote for or against publishing on a particular topic. Then the marked-up data is already transmitted to the neural network, which is trained on it and improved. Due to the large amount of marked-up data, the neural network is well trained and can find similar publications that are more likely to attract users' interest. One of the disadvantages is that not every project can attract a large number of users for data markup. In addition, this solution is not an open source solution.

Another example is Yandex music. The recommendation system analyzes the user's actions: likes and dislikes, skipped tracks, repeated playback, and so on. Each action has weights that are later used in the algorithm. In addition, the system analyzes similar profiles. The final list of recommendations is compiled using Matrixnet [5], which processes the list of all possible recommendations and determines which ones should be shown to the user on the Yandex Music home page and in what order to place them. It is worth noting that more than a hundred training models are used when making recommendations for a single user. This consumes a large amount of resources – hundreds of servers collect data about user requests to the search engine, viewed products, etc. this approach can be used by large companies, but it is not suitable for small projects.

It is worth noting that the systems described above and other similar systems are sharpened for a specific set of data that a particular service works with. Also, the entire data processing process (data cleaning, preprocessing, model learning) is not open source. This article will discuss the process of working with the most popular data types, as well as building an algorithm for data processing and training models in such a way that this algorithm can be reused on other data types and in other projects.

3. Approach to building a recommendation system

The data set analyzed in this article was collected in one of the photo hosting services. This data set contains 127 images, 307 comments, 496 rating entries (likes and dislikes), and 47 geotags. The recommendation system will consist of several data processing modules. The algorithm of the system is shown in fig. 1.

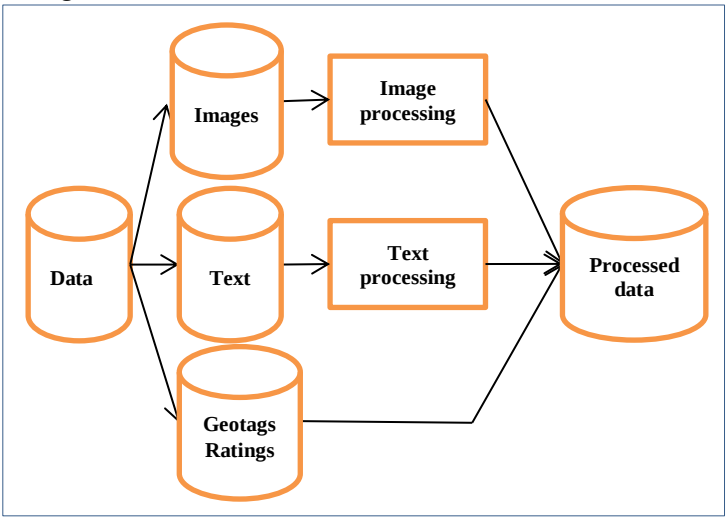


Fig. 1. The architecture of the recommendation system

Raw data is sent to the system input. This data is divided into three categories:

- images;
- text;
- geotags and ratings.

To identify user interests, images and text will be processed by machine learning modules. Image processing involves a module that will recognize objects in the image. Text processing includes two submodules: recognition of the main subject of the text and recognition of the tone of the text (positive or negative).

All processed data will be combined by a unique identifier (id). As a result, this will create a target tables that will contain the following information:

- what the user posts;
- what the user writes about and in what key;
- what the user evaluates positively;
- what the user evaluates negatively;
- geotags attached to the user's records.

This data will help you identify user groups based on their interests. You can use interest groups to recommend new publications, recommend various products, and so on.

It is worth noting that MySQL [6] relational database was chosen for storing information. Moreover, images are not stored directly in the database, but are stored in the file system. The database stores only links to images. Machine learning modules are written in Python, as this language offers a wide range of tools for data processing.

4. Machine learning modules

Let's take a closer look at how machine learning modules work for image and text processing.

4.1 Module for recognizing objects in an image

The pre-trained Inception-v3 [7] model was used for recognizing objects in images. This is one of the most popular models for recognizing objects in images [8]. This model achieves an accuracy of more than 78.1% on the Imagenet dataset. The model has been trained in 1000 [9] classes. The use of the pre-trained model is due to the fact that the model has good performance, has open source code, is easily integrated into existing solutions, and works fast enough (about 1-2 seconds for 1 image on Intel core i7).

When analyzing images, this model outputs the top prediction classes with the highest score value. Within the recommendation system, only the value with the highest score was recorded. An example of how the model works is shown in fig. 2:

```
flags =  Namespace(image_file='server_uploads/
30/8b10a257-7569-49c8-96e2-6d15676c9169.jpeg',
model_dir='tmp/imagenet', num_top_predictions=
5)
unparsed =  ['D:\\Users\\WinUser\\Anaconda3\\l
ib\\site-packages\\ipykernel_launcher.py', '-
f', 'C:\\Users\\WinUser\\AppData\\Roaming\\jup
yter\\runtime\\kernel-fafb8a94-6d99-4718-bbe4-
80870508319a.json']
television, television system (score = 0.8473
8)
```

Fig. 1. Example of how the image recognition model works

In total, 127 photos from the original data set were processed using this model. Of the 1000 classes available in the model, 87 images were recognized. The average score value for all data is about 0.49. Information about recognized objects is shown in fig. 3.

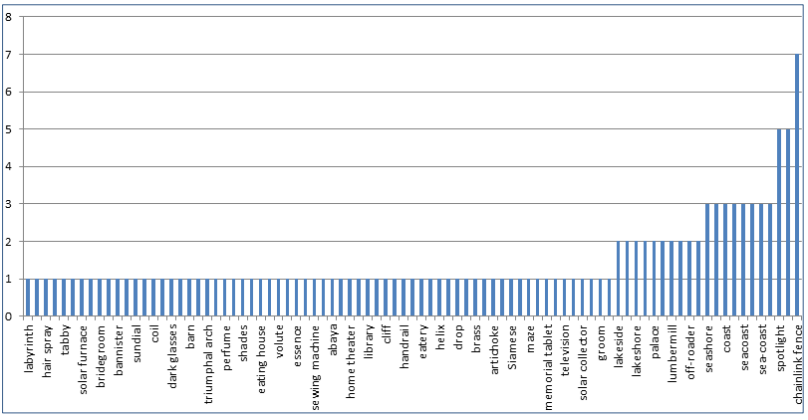


Fig. 2. Recognized objects

The most popular «chainlink fence» images shown on the chart are a classifier error, such images have a very small score. The most common objects are the sea, the coast, cars, and architectural objects. In the data set under consideration, the results of the classifier were analyzed. Correctly predicted values had a score greater than 0.5, so these images were considered correctly recognized and taken into account in the future (there are also incorrectly recognized images, but only about 10% of them).

All data was written to a MySQL table with the following fields:

- id;
- photo_id;
- photo_desc;
- score.

Here *id* is a unique identifier, *photo_id* is a foreign key from the photo table, *photo_desc* is the name of the recognized object, and *score* is the value of score.

4.2 The analysis module of text subject

Working with text is a more complex topic than image recognition, so there are no ready-made models here. This is because each language has its own grammar and it is difficult to adapt one model for all languages at once. In our case, the entire text was in Russian. However, there are various algorithms that can be adapted to your data and trained. To highlight the main topic of the text, a model based on the Latent Dirichlet Allocation (LDA) [10] algorithm was used. The main idea of this algorithm is that each document is considered as a set of topics in a certain proportion. Each topic is a set of the most common word and each document consists of a specific set of words [14].

The origin data set cannot be passed directly to the model. First, you need to additionally process the text:

- eliminate unnecessary characters (punctuation marks);
- remove stop words (conjunctions, particles, etc.);
- form stable phrases;
- make lemmatization.

The *simple_preprocess()* method of the Gensim library [11] was used to remove punctuation and tokenize the text. To delete stop words, a set of stop words from the *nltk* [12] package was used. Bigrams and trigrams were formed as stable phrases using the Gensim library. The *ru2* model from the spacy package was used for lemmatization.

The main input data for the LDA model is the dictionary and corpus. Gensim creates a unique identifier for each word in the document, and the corpus shows the frequency of occurrence of this word.

One of the hyperparameters is the number of topics in the text. Since we had a fairly small data set, we set 20 topics. Other alpha and eta (was set to 'auto') parameters affect the sparsity of topics, chunksize (was set to 100) – the number of documents in each training chunk, and passes (was set to 10) - the total number of training passes.

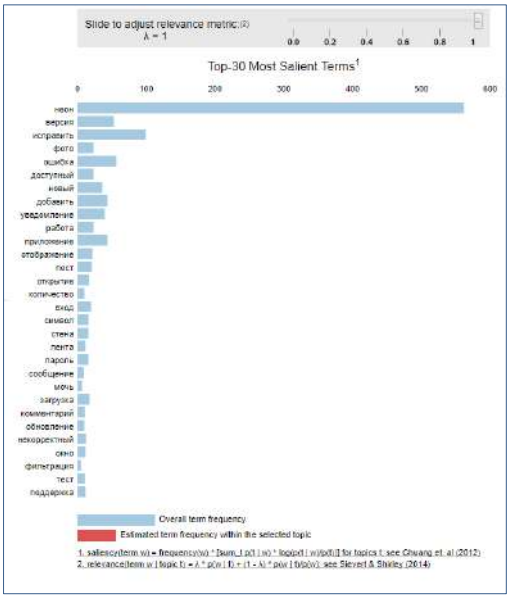


Fig. 3. Visualization of the LDA model operation

To visualize the result, an interactive diagram was built using the pyLDAvis [13] package, which is shown in fig. 4.

4.3 The analysis module of text sentiment

A convolutional neural network was used to analyze the tone of the text [15, 16, 17, 18]. The Word2Vec library was used to create the feature space. The training was conducted on a corpus of words based on Russian-language messages from Twitter, which contains 114991 positive and 111923 negative tweets, as well as 17639674 unmarked tweets [19]. Before training, all data was pre-processed (reduced to lowercase, replacing links to the token, etc.). The Word2Vec model was trained using the Gensim library. The Keras library [20] was used to build the neural network. This model, trained on tweets, was applied to text messages from the data set in question. The model metrics are shown in fig. 5.

	precision	recall	f1-score	support
0	0.76179	0.80437	0.78250	22236
1	0.79569	0.75180	0.77312	22534
accuracy			0.77791	44770
macro avg	0.77874	0.77808	0.77781	44770
weighted avg	0.77885	0.77791	0.77778	44770

Fig. 5. Metric models the tone of the text

This model was used to process the original data set that contained comments. As a result, the results were obtained as fig. 6 shows.

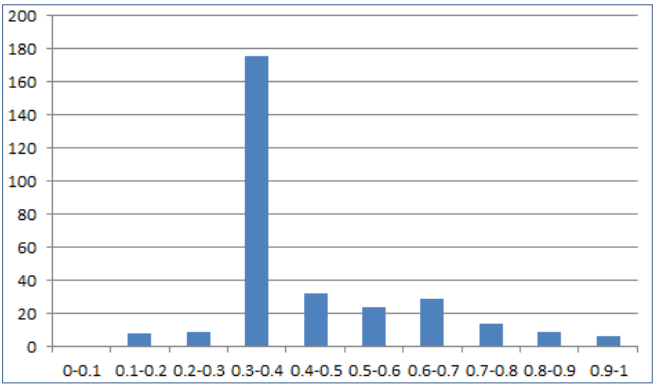


Fig. 6. The results of the model determine the tone of the text

In this case, the abscissus axis shows the percentage predicted by the model, and the ordinate axis shows the number of similar comments. As you can see, most of the comments in the provided data set had a mostly neutral accent (values between 0.3 and 0.7 were taken as neutral, this data was viewed manually).

The trained model was used on the source data. All results were written to a MySQL table.

5. Results

The results of all three models were recorded in MySQL. All data is combined with a single id. This way we can now distinguish user groups based on their interests. As a result, the database contains the following tables.

- Post. This table stores the id, photo and / or text, rating, geolocation (if available), and author of the publication;

- Comment. This table stores the id, publication id, text, rating, and comment author;
- Rating. This tables stores id, user id, photo id and rating (negative, positive or neutral);
- Object in the photo. This table stores the id, information about objects in the image (this information was obtained using the model), and the image id;
- Main theme of the text. This table stores the id, the main subject of the text, the type of post (post or comment), and the id of the post or comment.
- Tone of the text. This table stores the id, the tone of the text, the type of post (post or comment), and the id of the post or comment.

Let's look at an example of making recommendations using these tables. Let's say that we create an individual recommendation system to recommend interesting authors. To do this, we need to select what the user posts and what they rate positively (posts and comments). Then we need to find authors who publish similar images and recommend such authors to the user. For example, if these are sea coasts, we can use the following SQL query: «*select distinct (photo.userid) from photo_desc, photo where (photo_desc.photo_desc like '%coast%' or photo_desc.photo_desc like '%sea%') and photo_desc.photo_id=photo.id and score > '0.5'*». Three such users were found in the data set under consideration (fig. 7).

	userid
▶	45
	66
	99

Fig. 4. Result of the SQL query

6. Conclusion

As a result, we implemented a recommendation system that allows us to identify target groups of users. The process of data processing by several machine learning models was considered. The Concept-v3 model was used for image processing, an LDA-based model was used to highlight the subject of the text, and a neural network-based model was used to determine the tone of the text. The model results were used for building SQL queries. The results of all models in the test data set were checked manually. For the object recognition model, the extreme score value was set to 0.5. For the text tone recognition model, the values 0-0. 3 were set for negative text, 0.3-0.7 for neutral text, and 0.7-1 for positive text.

This system can be used on small projects, since models are trained on marked - up data from open sources. In addition, the logic of setting up search targets is quite clear; it can be performed by any analyst who knows the SQL language. This architecture is suitable for almost any purpose, whether it is recommending services, searching for interesting publications, etc.

In future plans:

- Building the process of fully automating the launch of model training. To do this, you plan to use the Linux scheduler, or Jenkins/TeamCity;
- Implementation of a recommendation system in a real project. At this point, the data was received as a separate set of values and processed on a separate computer. For the full operation of the service, it is planned to transfer the entire data processing process to an industrial server;
- Analysis of model metrics. After implementing this system in the service, it is planned to analyze the accuracy of the models. This can be tracked by user clicks on the proposed content. It will also allow you to conduct A / B tests when some users see suggestions of recommendations from one model, and others from another. These tests will help you identify the best-performing models.

References / Список литературы

[1]. Recommendation Technology ‘Disco’. URL: https://yandex.com/company/technologies/disco/.

[2]. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations. In Proc. of the 10th ACM Conference on Recommender Systems - RecSys ’16, 2016, pp. 191-198.

[3]. Gomez-Uribe C.A., Hunt N. The Netflix Recommender System. ACM Transactions on Management Information Systems, vol. 6, issue 4, 2015, pp. 1-19.
[4]. VK Experts. URL: https://vk.com/press/theme-feeds.
[5]. Matrixnet. URL: https://yandex.ru/company/technologies/matrixnet/.
[6]. Joel Murach. Murach's MySQL. Mike Murach & Associates, 2012, 612 p.
[7]. TensorFlow models, GitHub. URL: https://github.com/tensorflow/models.
[8]. Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the Inception Architecture for Computer Vision. In Proc. of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826.
[9]. 1000 synsets for Task 2 (same as in ILSVRC2012). URL: http://image-net.org/challenges/LSVRC/2014/browse-synsets.
[10]. Bíró I., Szabó J. Latent Dirichlet Allocation for Automatic Document Categorization. Lecture Notes in Computer Science, vol. 5782, 2009, pp. 430-441.
[11]. Gensim project page. URL: https://pypi.org/project/gensim/.
[12]. NLTK project page. URL: https://www.nltk.org/.
[13]. pyLDAvis project page. URL: https://www.nltk.org/.
[14]. Thematic modeling using Gensim (Python). URL: https://webdevblog.ru/tematicheskoe-modelirovanie-s-pomoshhju-gensim-python/.
[15]. Jin R., Lu L., Lee J., Usman A. Multi-representational convolutional neural networks for text classification. Computational Intelligence, vol. 35, issue 3, ,2019, 599–609.
[16]. Text tonality analysis using convolutional neural networks. URL: https://habr.com/ru/company/mailru/blog/417767/.
[17]. Cliche M. BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs. In Proc. of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 573-580.
[18]. Zhang Y., Wallace B.A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820, 2015.
[19]. Rubtsova Y.V. (2015). Constructing a corpus for sentiment classification training. Programmnye produkty i sistemy, no. 27, 2015, pp. 72-78 (in Russian) / Рубцова Ю.А. Построение корпуса текстов для настройки тонового классификатора. Программные продукты и системы, no. 27, 2015 г., стр. 72-78.
[20]. Keras project page. URL: https://keras.io/.

Information about authors / Информация об авторах

Vitaly Viktorovich MONASTYREV – student. Research interests: neural networks, recommender systems, machine learning.

Виталий Викторович МОНАСТЫРЕВ – студент. Научные интересы: нейронные сети, рекомендательные системы, машинное обучение.

Pavel Dmitrievich DROBINTSEV – Ph.D., Associate Professor. Research interests: test automation, formal models, software verification, artificial intelligence applications.

Павел Дмитриевич ДРОБИНЦЕВ – к.т.н., доцент. Научные интересы: автоматизация тестирования, формальные модели, верификация программного обеспечения, приложения искусственного интеллекта.

DOI: 10.15514/ISPRAS-2020-32(3)-10

Machine Learning-Based malicious users' detection in the VKontakte social network

D.I. Samokhvalov, ORCID: 0000-0003-4719-9638 <disamokhvalov@edu.hse.ru>
National Research University Higher School of Economics,
20 Myasnitskaya ulitsa, Moscow, 101000 Russia

Abstract. This paper presents a machine learning-based approach for detection of malicious users in the largest Russian online social network VKontakte. An exploratory data analysis was conducted to determine the insights and anomalies in a dataset consisted of 42394 malicious and 241035 genuine accounts. Furthermore, a tool for automated collection of the information about malicious accounts in the VKontakte online social network was developed and used for the dataset collection, described in this research. A baseline feature engineering was conducted and the CatBoost classifier was used to build a classification model. The results showed that this model can identify malicious users with an overall 0.91 AUC-score validated with 4-folds cross-validation approach.

Keywords: VKontakte; malicious users; machine learning; social networks; classification models

For citation: Samokhvalov D.I. Machine Learning-Based malicious users' detection in the VKontakte social network. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 109-118. DOI: 10.15514/ISPRAS–2020–32(3)–10

Определение аккаунтов злоумышленников в социальной сети ВКонтакте при помощи методов машинного обучения

Д.И. Самохвалов, ORCID: 0000-0003-4719-9638 <disamokhvalov@edu.hse.ru>
Национальный исследовательский университет «Высшая школа экономики»,
Россия, 101000, г. Москва, ул. Мясницкая, д. 20

Abstract. В данной работе представлен подход для обнаружения аккаунтов злоумышленников в крупнейшей российской социальной сети ВКонтакте на основе методов машинного обучения. Был проведен исследовательский анализ данных для определения аномалий и закономерностей в наборе данных, состоящем из 42394 вредоносных и 241035 подлинных учетных записей пользователей ВКонтакте. Кроме того, для получения набора данных был разработан инструмент для автоматического сбора информации о вредоносных аккаунтах в социальной сети ВКонтакте, описание архитектуры данного инструмента приведено в работе. На основе признаков, сгенерированных из пользовательских данных, была обучена модель классификации при помощи библиотеки CatBoost. Результаты показали, что эта модель может идентифицировать злоумышленников с общим качеством AUC 0.91, подтвержденной четырехкратным методом перекрестной проверки.

Ключевые слова: ВКонтакте; злоумышленники; машинное обучение; социальные сети; модели классификации; анализ данных

Для цитирования: Самохвалов Д.И. Определение аккаунтов злоумышленников в социальной сети ВКонтакте при помощи методов машинного обучения. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 109-118 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–10

1. Introduction

An online social network (OSN) is an online platform that allows people who share the same views or have real-life connections to interact with each other online [1]. OSNs also provide users with a great ability to communicate, entertain, consume and share a different type of information that they are interested in. Moreover, modern social networks have become the platforms where companies can promote and even sell their products while maintaining good relationships with their customers through clear communication channels [2, 3].

Being a great instrument for connecting people and hosting useful information, OSNs try to attract as many users as possible, thus a strong authentication (by personal ID or driving license for ex.) is not required for an account creation as a rule. For example, in the OSN VKontakte, for a long time, it was possible to register an account by submitting only an e-mail address. VKontakte team made an authentication by mobile telephone number required for a valid account creation, however, this still does not fully solve the issue, since it is possible for a malicious identity to use multiple sim-cards or so-called virtual numbers [4].

Lack of strong authentication provides an opportunity for malicious users to evade OSNs with malicious activity, such as spamming, phishing, distribution of malicious software, trolling, terrorism and others [5-8]. While these are the activities that evaded the internet almost since its invention, several new threats relevant to OSNs have appeared [9. 10].

- *Clickjacking* – a malicious practice where a user is made to click on something that behaves not the same way as it should to the prior knowledge of the user.
- Crowdturfing – a campaign that aims to gain or destroy the reputation of people, products and other entities through spreading biased opinions and framed information.
- Fake account attack – a most commonly used type of attack when an account with fake credentials created for interaction with the legitimate users.
- *Identity clone attack* – a malicious practice where an attacker creates a new fake profile while using stolen private information of an existent user.
- *Cyberstalking* – harassment of an individual in the social network.

The aforementioned threats are relevant for most of the existing social networks and in most cases, they are performed by fakes.

Facebook, the largest social network in the world, reports that 8.7\% of its accounts which amounts to approximately 206 million do not belong to real users [11]. For addressing this vital issue Facebook even created its security system for protecting users from malicious activity and it is known as Facebook Immune System (FIS) [12]. While being a scalable real-time system that can process hundreds of thousands read and write actions per second, it cannot still detect all the types of malicious activity [13, 14].

The goal of this research is to analyze the application of machine learning techniques for the detection of malicious users in OSN VKontakte. The information about the total number of 42394 malicious accounts was collected with the help of developed automated VK-scraper tool. In this research, we show that VKontakte malicious users have a specificity that is possible to use for building a highly accurate classification model.

The main contributions of this paper are the following.

- We propose an architecture for automated malicious accounts collection tool called VK-scraper.
- An exploratory data analysis of malicious VKontakte accounts was conducted and the main differences between malicious and genuine accounts were revealed.
- We show that Catboost performs better than Neural Nets approach proposed by other researcher for this problem.
- We provide a benchmark of the most important features identified by Catboost.

The outputs of this paper can be used further by other researches of malicious activity in VKontakte OSN.

2. Related work

The machine learning-based detection of malicious users in OSNs has attracted the attention of both researches and businesses when machine learning became an industrially popular and valuable approach. In [15] an application of Matrix factorization and SVM for spam accounts detection in Chinese OSN Renren was proposed. In this work, authors collected a dataset out of 33116 accounts, manually classified them into spammers and non-spammers and applied the SVM algorithm for spammers detection on a set of messages content and users' social behavior. They managed to reach an outstanding performance with a true positive rate of spammers detection reaching 99.1\%.

The Longitudinal Data Analysis of the Social graph method for the detection of so-called «Friends farms» in VKontakte was developed in [16]. This work aimed to detect fake identities among newly registered users of vk.com. According to conducted longitudinal analysis, authors revealed that fake profiles are more likely to be found among those users that show abnormal behavior in the growth of social graph metrics such as degree, reciprocated ties and clustering.

In [14] a framework for detecting Fake account attacks on Facebook was described. The research studied the temporal evolution of OSNs and the characteristics of the real users' profiles. Researchers presented a way to analyze social network graphs from a dynamic point of view within the context of privacy threats.

The application of machine learning techniques for fake profiles identification in Linkedin was described in [17]. Since Linkedin is a quite closed OSN that does not expose any API to the outer world, it is rather hard to get any data for the analysis from there. Authors of this work showed that even having a very limited dataset of only 27 fake accounts, it is possible to achieve a result comparable to the results obtained by other existing approaches based on the larger data set.

An instrument called SybilRank was developed in [18]. SybilRank is used for detecting the fake users (called Sybils) in Tuenti OSN by analyzing the social graph properties. The developed tool allowed to achieve at least 20\% lower false positive and negative rates than the second-best contender in most of the attack scenarios.

Sophisticated techniques for data normalization and noise removals such as Artificial Bee Colony (ABC) and Ant Colony Optimization (ACO) were used in [19] among which 3 supervised machine learning algorithms (Naive Bayes, SVM, and Decision Trees) were applied to predict the fake users' profiles on Facebook.

The CRAWLER tool was developed in [20] and a total number of 992 profiles were crawled with the help of this tool, out of which 201 turned out to be malicious. An application of both supervised (Decision Trees, KNN, SVM) and unsupervised (K-means, K-medoids) machine learning algorithms were used for classification, and a decent qualities of the models were obtained.

In [21], an application of methods such as PCA, Spearman's Rank-Order Correlation, Wrapper Feature Selection using SVM is described for dimensionality reduction to reduce the number of low-importance features for the fake accounts' detection in the social media. In the research, several existing datasets of both real and fake Twitter accounts, crawled by other researchers, is used. A set of feature selection techniques was evaluated to achieve the best performance and classification results.

In [22], an analysis of the tonality of the statuses of users of the OSN Facebook is conducted. Authors compared machine learning algorithms Naïve Bayes, Rocchio, and multi-layer perceptron by applying them on the 7000 status updates received from 900 Facebook users. All of the statuses were manually divided into two classes: positive and negative, however since there were significantly fewer negative reviews in the sample, the authors used 1131 reviews of each class to balance the classes in the final training dataset.

In [12], a software application and architecture described. The application aims to protect users and the social graph from malicious actions by cybercriminals. The described system operates in real-time and, according to the statements of its creators, checks and classifies each read and write action. As of March 2011, the system performed 25 billion checks per day, with a peak frequency of 650,000 checks per second.

Authors of [23] describe an approach to identify automatically managed accounts or so called bots in the VKontakte OSN. Authors use a feedforward neural network and a sample of 4918 blocked accounts to train the model that shows a decent result on the validation set. Authors use an approach for sampling malicious accounts that is similar to one described in this paper, however the method they use in their research is not automated and thus can not be done in a standalone way. There is now evidence of what features turned out to be the most important and also it is not clear how exactly status-based features were generated.

In [24], authors explore stacking ensemble approach on top of a combination of different types of models that were trained on the attributes of three different types: friendship graph, subscription information and user's texts. The result received in this article is 4-9\% better than in [23].

In [25], a framework for extracting a large collections of Twitter accounts was proposed. Based on these features, several highly accurate models were built and their performances were evaluated on both an existing public dataset and an additional sample of manually-annotated Twitter accounts collected with a different strategy. Based on the models predictions, authors evaluated that percentage of Twitter accounts exhibiting social bot behaviors is between 9\% and 15\% and the behaviour of such accounts can be detected by supervised machine learning techniques.

In [26], a model which increases the recall in detecting bots, allowing a researcher to delete more bots in Twitter, was proposed. Authors proposed an algorithm called Boosting through Optimizing Recall which was applied on top of a combination of twitter-specific heuristic features and features obtained through topic modelling of the tweets. The algorithm showed a result relatively better than other state-of-the-art models like AdaBoost.

3. Proposed method

In this paper, a description of state-of-the-art machine learning techniques application for malicious users' identification in VKontakte OSN is presented. Moreover, an automated tool VK-scraper for scraping the data about malicious accounts before their actual removal by VKontakte administration is developed and its architecture is described in this research. A sample of 42395 of actual malicious users was collected and a set of data and feature engineering techniques were applied before the actual ML-model training.

3.1 VK-scraper

One of the most challenging parts of the malicious accounts detection domain is data collection. Even though some OSNs provide a useful API for the developers to interact with the platform and query the publicly available data, there is still a lack of techniques that allow gathering the available information about the blocked accounts since this data is not exposed by OSNs to the outer world after an account was blocked for a malicious activity. There were some workarounds proposed by researchers to deal with this obstacle, for example, expert evaluation, manual labeling, friends connections crawling, social graph properties analyses, etc. [16, 17, 20].

As was noted in [17] VKontakte assigns a unique incremental id to every user that is registered on the platform, thus it is easy to reverse engineer the relative timeline of VKontakte accounts registration. Since most of the malicious accounts are manually banned by VKontakte administration (due to the legitimate users complains mostly) within the first week of their existence, it is quite hard to detect a malicious user among the users that were registered a long time ago.

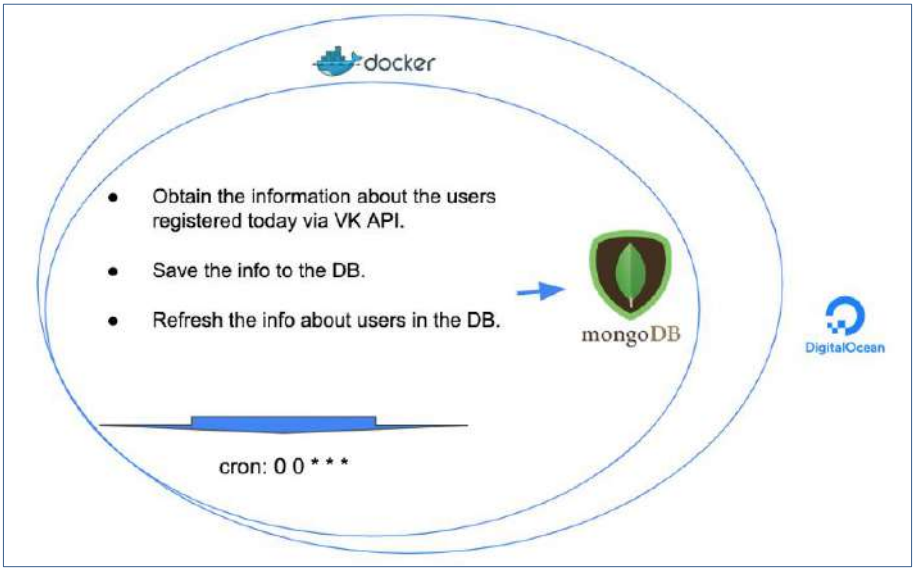


Fig. 1. VK-Scraper architecture

VK-scraper tool works in the following way (fig. 1). Every day it checks if there were any changes in the data that are stored in the VK-scraper MongoDB [27] database by simply calling the VKontakte API and comparing the data from the response to the data stored in the database. If there was a change, for example, a user updated its status or has been banned by the administration, it updates the information in the database by changing the differing fields. After that, it collects the information about 120,000 newly registered accounts in VKontakte by simply iterating over the 120,000 largest accounts ids that exist in the OSN. The newly scraped ids are stored in the VK-scraper database.

MongoDB was used as a local DB for storing data as it perfectly suits for storing JSON data and does not require a schema.

VK-scraper is wrapped with Docker [28] and deployed on a dedicated VPS provided by DigitalOcean developer cloud [29].

VK-scraper worked for 30 days (from 01.10.2019 to 30.10.2019) on a dedicated VPS and collected information about 3.5 million accounts, out of which 42394 turned out to be malicious.

3.2. Feature Engineering

VKontakte API provides access to query all the publicly available information about any open VK account. For example, it is possible to get information about the schools or universities that a specific user attended or what types of music she prefers if this data is provided by the user. Most of the accounts features available via VK API are categorical. The categorical feature is a feature that has a discrete set of values that are not necessarily comparable with each other (e.g., user ID or name of a city) [30].

Unfortunately, the number of values that are relevant for some feature can be quite large (for example, there are more than 200 countries available for selection during registration in vk.com) and this can make the model training and evaluation quite hard and even biased if the training dataset is limited and cannot cover all the available values. Thus, unlike other approaches specified in [23, 24], it was decided to convert all the categorical features into binary which are simply the indicators of whether this feature was specified by the account holder or not.

3.3 Catboost

There are plenty of machine learning algorithms for solving a binary classification model available today. One of the most robust is gradient boosting algorithm. Catboost [30] – is an open-source library developed by Russian tech-giant Yandex that implements gradient boosting algorithm with special orientation on performance and processing of categorical features. It outperforms other popular implementation of gradient boosting in terms of quality on the classification tasks.

4. Experiments and results

Before building the actual model, an Exploratory Data analysis was conducted to compare the malicious and genuine user datasets and find the anomalies or extract the insights from the data. After that, a CatBoost model was trained on 4-fold cross-validation with the Log-Loss metric optimized on the fly.

4.1 Exploratory data analysis

A comparison of malicious and genuine accounts dataset revealed that there is a larger portion of genuine users who has certain info fulfilled in their profiles rather than malicious users (fig. 2). For example, 57% of genuine users specified the country they currently live in their profiles, compared to 28% for malicious accounts; 40% of genuine users indicated the schools they studied in, while only 15% of malicious accounts had this information in their profiles. Most of the malicious accounts (78%) have female sex and also most of them (81%) have at least one photo uploaded. 36% of malicious accounts has at least one friend. Two most popular professions are entrepreneur and princess. It was also revealed that 98% of malicious users have their mobile phones connected to their accounts, while only 59% of genuine users linked their phone numbers to their profiles.

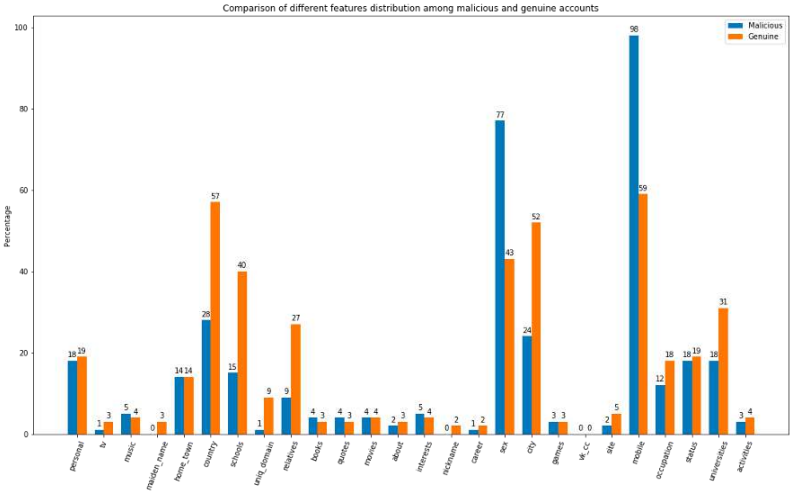


Fig. 2. Comparison of different features distribution among malicious and genuine accounts

After researching for a while about that, it was found out that until a certain time, it was possible to register in VKontakte without having a phone number linked to the account during the registration process, however, nowadays it is impossible to create an account without having a mobile phone number assigned to the actual account. Since the malicious users' dataset contains only newly registered users and the genuine dataset is by one half a random selection from all of the existing accounts, it was decided to remove this feature from the both of the datasets when training the model to prevent overfitting on the peculiar properties of our data.

Another interesting part that requires a more detailed exploration is user statuses. Status - is a short (less than 140 symbols) text that a user can outline right below his profile name on the main page.

It was found out that roughly 19% of malicious accounts have a status specified on their page and the meaning of those statuses is generally malicious. The biggest portion of the accounts promote sexual services or contain extremist slogans. It might be interesting to research on NLP techniques application for detection of malicious statuses in the future.

4.2 Model training

The CatBoost classifier was trained on the given dataset with 4-fold cross-validation, with 20% of the dataset belonging to a fold on each training round (Fig. 4).

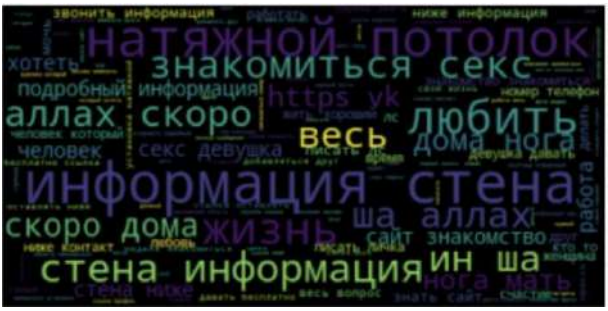


Fig. 3. Malicious users' statuses wordcloud.
The larger the word the more often it occurs in the statuses of malicious accounts

As can be seen, a baseline of 0.91 was achieved on 4-fold cross-validation which is 15 points better than in [23]. A benchmark of feature importance is provided in Table 1.

Table 1. Feature importance

Feature	Importance
bdate-day	22.335
sex	15.46
has-country	12.638
has-schools	8.989
has-relatives	8.532
friends-number	7,648
unique-domain	2.738
has-city	1.674
has-maiden-name	1.026
has-status	0.417

5. Further directions

Further research concerns both technical and performance improvement of the VK-scraper tool and investigation of more advanced supervised learning techniques such as Deep Learning and NLP. It also might be interesting to try to apply some rotating proxy services such as Micro-leaves [31] for bypassing VKontakte API query restrictions and a larger-scale network scanning. This approach might require a larger-scale computational resource.

The application of Deep Learning models can be considered after a larger sample of malicious users will be gathered. For example, it could be interesting to predict the probability of an account being malicious by applying Convolutional Neural Networks to profile photos of the gathered accounts. Another interesting direction is the application of NLP techniques for analyzing the accounts statuses and wall posts. It might be not such a good standalone solution since the majority of accounts do not have any status in their profiles and the daily quota for the wall API requests is limited to 5000 per

day, but within an ensemble of Machine Learning and Deep Learning models, it could make a valuable contribution to the overall ensemble score.

6. Conclusion

In this work, a Machine-Learning based approach for detection of the malicious users in the VKontakte OSN was presented. 42394 malicious users were scraped with the developed automated tool called VK-scraper. An exploratory data analysis for both malicious and genuine datasets was conducted and revealed that there is an evident difference between malicious and genuine VKontakte accounts.

Table 2. 4-fold cross-validation AUC scores

	f0l	f1l	f2l	f3l	f0t	f1t	f2t	f3t
curr.	0.89	0.89	0.89	0.89	0.88	0.89	0.89	0.89
best.	-	-	-	-	0.90	0.91	0.91	0.91

While the result of 0.91 AUC-score (Table 2) looks promising, there is still a room for improvement where more sophisticated techniques such as Deep Learning and NLP might come in.

References / Список литературы

[1]. J.A. Obar and S.S. Wildman. Social Media Definition and the Governance Challenge: An Introduction to the Special Issue. Telecommunications Policy, vol. 39, no. 9, 2915, pp. 745-750

[2]. D. M. Romero, W. Galuba, S. Asur, and B. A. Huberman. Influence and passivity in social media. In Proc. of the 20th International Conference Companion on World wide web, 2011, pp. 113-114.

[3]. Дубль [1] J. A. Obar and S. Wildman, "Social media definition and the governance challenge: An introduction to the special issue," Telecommunications Policy, vol. 39, no. 9, pp. 745–750, Oct. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0308596115001172

[4]. I. Shatilin. What are virtual SIM cards and what do they do? Available at: https://www.kaspersky.com/blog/virtual-sim/11572/.

[5]. K. S. Adewole, N. B. Anuar, A. Kamsin, K. D. Varathan, and S. A. Razak. Malicious accounts: Dark of the social networks. Journal of Network and Computer Applications, vol. 79, 2017, pp. 41–67.

[6]. A. V. Filimonov, A. V. Osipov, and A. B. Klimov. Application of neural networks to identify trolls in social networks. arXiv:1504.07416 [cs], Apr. 2015.

[7]. A. Malm, R. Nash, and R. Moghadam. Social Network Analysis and Terrorism. In Handbook of the Criminology of Terrorism, G. LaFree and J. D. Freilich, eds., John Wiley & Sons, Inc., 2017, pp. 221–231.

[8]. Z. Mao, D. Li, Y. Yang, X. Fu, and W. Yang. Chinese DMOs' engagement on global social media: examining post-related factors. Asia Pacific Journal of Tourism Research, vol. 25, no. 3, pp. 274–285.

[9]. D. DeBarr and H. Wechsler. Using Social Network Analysis for Spam Detection. Lecture Notes in Computer Science, 2010, vol. 6007, pp. 62–69.

[10]. L. Wu and H. Liu. Detecting Crowdturfing in Social Media. In Encyclopedia of Social Network Analysis and Mining, R. Alhajj and J. Rokne, eds, Springer, 2017, pp. 1–9.

[11]. M. Fire, D. Kagan, A. Elyashar, and Y. Elovici. Friend or foe? Fake profile identification in online social networks. Social Network Analysis and Mining, vol. 4, 2014, Article no. 194

[12]. T. Stein, E. Chen, and K. Mangla. Facebook immune system. In Proc. of the 4th Workshop on Social Network Systems, 2011, article no. 8, pp, 1–8 pp. 1–8.

[13]. S. Ali, N. Islam, A. Rauf, I. Din, M. Guizani, and J. Rodrigues. Privacy and Security Issues in Online Social Networks. Future Internet, vol. 10, no. 12, 2018, article no. 114, pp. 1-12.

[14]. M. Conti, R. Poovendran, and M. Secchiero. FakeBook: Detecting Fake Profiles in On-Line Social Networks. In Proc. of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2012, pp. 1071–1078.

[15]. A.J. Banu, N.N. Ahamed, B. Manivannan, K. Vanitha, M.M. Musthafa. Detecting Spammers on Social Networks. International Journal of Engineering and Computer Science, vol. 6, issue 2, 2017, pp. 20240-20247.

[16]. A. Romanov, A. Semenov, and J. Veijalainen. Revealing Fake Profiles in Social Networks by Longitudinal Data Analysis. In Proc. of the 13th International Conference on Web Information Systems and Technologies., 2017, pp. 51–58. 8
[17]. S. Adikari and K. Dutta. Identifying fake profiles in linkedin. In Proc. of the 19th Pacific Asia Conference on Information Systems, 2014, article no. 278.
[18]. Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In Proc. of the 9th USENIX Conference on Networked Systems Design and Implementation, 2012, pp. 197–210.
[19]. S. Y. Wani, M. M. Kirmani, and S. I. Ansarulla. Prediction of fake profiles on facebook using supervised machine learning techniques-a theoretical model. International Journal of Computer Science and Information Technologies, vol. 7, no. 4, 2016, pp. 1735–1738.
[20]. M. Albayati and A. Altamimi. MDFP: A Machine Learning Model for Detecting Fake Facebook Profiles Using Supervised and Unsupervised Mining Techniques. International Journal of Simulation: Systems, Science & Technology, vol. 20, no. 1, 2019, article no. 11, pp. 1-10.
[21]. S. Khaled, N. El-Tazi, and H.M.O. Mokhtar. Detecting Fake Accounts on Social Media. In Proc. of the 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 3672–3681.
[22]. C. Troussas, M. Virvou, K. J. Espinosa, K. Llaguno, and J. Caro. Sentiment analysis of facebook statuses using naive bayes classifier for language learning. In Proc. of the International Conference on Information, Intelligence, Systems and Applications, 2013, pp. 1–6.
[23]. P.D. Zegzhda, E.V. Malyshev, and E.Y. Pavlenko. The use of an artificial neural network to detect automatically managed accounts in social networks. Automatic Control and Computer Sciences, vol. 51, no. 8, 2017, pp. 874–880.
[24]. K. Skorniakov, D. Turdakov, and A. Zhabotinsky. Make social networks clean again: Graph embedding and stacking classifiers for bot detection. In Proc. of the 2nd International Workshop on Rumours and Deception in Social Media, 2018, paper 39.
[25]. O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini. Online human-bot interactions: Detection, estimation, and characterization. arXiv:1703.03107, 2017.
[26]. F. Morstatter, L. Wu, T. H. Nazer, K. M. Carley, and H. Liu. A new approach to bot detection: Striking the balance between precision and recal. In Proc. of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2016, pp. 533–540.
[27]. MongoDB, 2020. Available at: https://www.mongodb.com/
[28]. Docker, 2020. Available at: https://www.docker.com/
[29]. DigitalOcean, 2020. Available at: https://www.digitalocean.com/
[30]. L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. In Proceedings of the 32nd International Conference on Neural Information Processing, 2018, pp. 6638–6648.
[31]. Microleaves, 2020. Available at: https://microleaves.com/

Information about authors / Информация об авторах

Denis Igorevich Samokhvalov – master student. Research interests: social network analysis, machine learning.

Денис Игоревич Самохвалов – студент магистратуры. Научные интересы: анализ социальных сетей, машинное обучение.

DOI: 10.15514/ISPRAS-2020-32(3)-11

Development of automated computer vision methods for cell counting and endometrial gland detection for medical images processing

[1] *D.I. Sergeev, ORCID: 0000-0003-2503-6272 <densvr1@gmail.com>*
[2] *A.E. Andreev, ORCID: 0000-0003-3343-2937 <alexander597@mail.ru>*
[3] *A.O. Drobintseva, ORCID: 0000-0002-6833-6243 <anna.drobintseva@gmail.com>*
[1] *S. Cenevska, ORCID: 0000-0002-2272-8882 <slobodankacenevska@yahoo.com>*
[1] *N. Kukavica, ORCID: 0000-0001-6477-357X <nikola.kukavica.94@gmail.com>*
[1] *P.D. Drobintsev, ORCID: 0000-0003-1116-7765 <drob@ics2.ecd.spbstu.ru>*

[1] *Peter the Great St.Petersburg Polytechnic University,*
29, Polytechnicheskaya, St.Petersburg, 195251, Russia
[2]*The Research Institute of Obstetrics, Gynecology and Reproductology named after D.O. Ott,*
3, Mendeleevskaya line, Saint Petersburg, 199034
[3] *St.Petersburg State Pediatric Medical University*
2 Litovskaya st., St. Petersburg, 194100, Russia

Abstract. Current work is focused on the processing of medical images obtained by performing a pathomorphological analysis of preparation. The algorithms for processing images of nuclei of light and confocal microscopy and tissue of light microscopy were considered in particular. The application of the proposed algorithms and software for detecting pathologies was justified.

Keywords: computer vision; image processing; digital pathology; detection and classification; cell and tissue nuclei; light microscopy; confocal microscopy

For citation: Sergeev D.I., Andreev A.E., Drobintseva A.O., Cenevska S., Kukavica N., Drobintsev P.D. Development of automated computer vision methods for cell counting and endometrial gland detection for medical images processing. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020, pp. 119-130. DOI: 10.15514/ISPRAS–2020–32(3)–11

Разработка автоматизированных алгоритмов компьютерного зрения для обработки медицинских изображений

[1] *Д.И. Сергеев, ORCID: 0000-0003-2503-6272 <densvr1@gmail.com>*
[2] *А.Е. Андреев, ORCID: 0000-0003-3343-2937 <alexander597@mail.ru>*
[3] *А.О. Дробинцева, ORCID: 0000-0002-6833-6243 <anna.drobintseva@gmail.com>*
[1] *С. Ценевска, ORCID: 0000-0002-2272-8882 <slobodankacenevska@yahoo.com>*
[1] *Н. Кукавица, ORCID: 0000-0001-6477-357X <nikola.kukavica.94@gmail.com>*
[1] *П.Д. Дробинцев, ORCID: 0000-0003-1116-7765 <drob@ics2.ecd.spbstu.ru>*

[1] *Санкт-Петербургский политехнический университет Петра Великого, 195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29*
[2] *НИИ акушерства, гинекологии и репродуктологии имени Д.О. Отта, 199034, г. Санкт-Петербург, Менделеевская линия, 3*
[3] *Санкт-Петербургский педиатрический университет, 194100, Россия, Санкт-Петербург, ул. Литовская, д.2*

Abstract. Текущая работа ориентирована на обработку медицинских изображений, полученных путем проведения патоморфологического анализа препарата. В частности, были рассмотрены алгоритмы обработки изображений ядер световой и конфокальной микроскопии и изображений тканей световой микроскопии. В работе доказана применимость предложенных алгоритмов и программного обеспечения для выявления патологий.

Ключевые слова: компьютерное зрение; обработка изображений; цифровая патология; обнаружение и классификация; ядра клеток и тканей; световая микроскопия; конфокальная микроскопия

Для цитирования: Сергеев Д.И., Андреев А.Е., Дробинцева А.О., Ценевска С., Кукавица Н., Дробинцев П.Д. Разработка автоматизированных алгоритмов компьютерного зрения для обработки медицинских изображений. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 119-130 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(3)–11

1. Introduction

The processing of medical images is an extremely important issue for biology and medicine. Pathomorphologists have to process hundreds of images of preparations per day. Their work can be automated due to computer analysis.

Medical image processing can be performed in semi-automatic and automatic modes. Semi-automatic mode is based on manual adjustment of simple, intuitive parameters for evaluating single microphotographs. Automatic mode otherwise does not require both direct operator intervention and initial settings in the processing of preparations.

Modern experts in the field of pathomorphology have an access to a wide range of technologies that make it possible to carry out various measurements depending to required tasks. A striking example of such technologies is the universal ImageJ software [1], which is capable for performing operations aimed at evaluating the geometry and color gamut of the resulting images. The main idea of this software is to write macroses that require a minimum understanding of computer technology from a specialist. This approach makes ImageJ flexible, but not user friendly. More intuitive tools are commercial software such as VideoTestMorphology [2, 3] and ImageProPlus [4]. There are various microcopy solutions. The most popular software at the moment are compared in Table 1.

Table 1. Software features

Software	Language	Automatic mode	Machine learning	Coding	Open source
Image J [1]	Java	macros	plugin	no	no

Cell profiler [5]	Python	macros	no	yes	no
Orbit Image Analysis [6]	Java, python	yes	yes	no	yes
Axio vision 4.8 [7]	-	macros	yes	no	no
Video test morphology 5.2 [3]	-	macros	yes	no	no
CellSens [8]	-	macros	no	no	no
IMAGE-PRO-Premium [4]	.NET	yes	yes	no	no
BioVision	-	yes	yes	no	no

Thus, there are a lot of modern software, designed for morphometry. However, there are a number of facts that severely limit the domain specialist in choosing his own tools. Firstly, most of the free automatic and semi-automatic programs are incomplete and require bioinformatics in the team. Secondly, the already collected, semi-automatic programs in most cases are universal, which means the absence in the process of taking into account the parameters of the microscope, the type of tissue being examined and the lighting of a particular picture. These parameters must be driven manually, relying on the empirical experience of the specialist responsible for setting up the program. Thirdly, the most convenient and user-friendly programs have a high cost. Therefore, they are not available in small laboratories.

The aim of this work is the development of software that partially eliminate the shortcomings of modern non-commercial software for processing digital images with automatic objects recognition by series of images and extracting from them the minimum set of basic features necessary for researchers to work with.

Typical operations for this task are initialization, localization, segmentation, shape analysis, modeling, analysis of cell parameters, etc. [9]. Although there are many methods of segmentation, precise segmentation is a complex task, and it plays a significant role in biological imaging studies

2. Typical tasks

2.1 Processing of cell and tissue preparations

Processing of preparations is carried out both at the cellular and tissue levels. In both cases, the cells are usually tinted with the help of their special reaction to the examined «marker». In this study, nuclear markers such as estrogen receptors (ER) and progesterone (PR) were detected. A quantitative analysis of nuclei with receptors for ER and PR is essential, since they are involved in the mechanisms of growth and metastasis of tumors. The research of the expression of ER and PR is included in the standard of examining patients with breast cancer, as it allows us to determine the sensitivity of the tumor to hormone therapy and to clarify the prognosis of the disease.

The research of ER and PR is also used in the diagnosis of infertility, endometrial hyperplasia [10].

Pathomorphologist needs to calculate the number of cell nuclei highlighted in color on the preparation, which correspond to the expression of the researched markers, as well as the total number of nuclei per unit area.

Another assignment of the pathomorphologist is to isolate the contours of glands and tissues on the preparation and determine the number of glands with high total marker expression.

According to statistical data, such as the number of nuclei and glands with and without marker expression, conclusions about the structure of the tissue are made, the effectiveness of treatment is considered, a diagnosis and the prognosis of the disease are specified.

2.2 Light and confocal microscopy

Medical preparations can be obtained in various ways. Preparations obtained using light and confocal microscopes are studied in this work. Confocal microscopy has been used relatively recently and, compared to the light one, gives more contrasting color images, with staining of the nuclei with different fluorescent dyes with antibodies of the corresponding markers. However, confocal microscopy requires special equipment and the quality of the preparation, which makes it impossible for mass application. Therefore, for the diagnosis of diseases light microscopy is still used in most cases [11].

3. Popular algorithms

Medical imaging algorithms typically consist of the steps described in this section below.

3.1 Image pre-processing

It is used to create conditions that increase the efficiency and quality of the isolation and recognition of nuclei in medical preparations. It includes morphological operators and filters, border detectors, filters with brightness normalization [12].

3.2 Detection of objects of interest

At this stage, the X and Y coordinates of the proposed center of each object of interest (nucleus) are determined. As a result of the stage, a set of objects of interest, which are probably nuclei is obtained. The algorithm parameters at this step are set in the particular way in order to create the redundant number of objects of interest. In other words, the detection of false objects is acceptable, but the admission of real cores is not. Basically, the following algorithms are used: active contour algorithm, watershed algorithm, image segmentation by known classes, segmentation with preliminary detection of class boundaries [12].

3.3 Selection of characteristics of objects of interest, classification and arrival at a decision

The final stage allows to attribute each of the objects of interest to one of the target classes. For the task of classifying nuclei using fluorescence and light microscopy preparations, the topological, texture, and color intensity-based characteristics of following classes are distinguished.

At the classification stage, objects of interest classes are determined by markers used for coloring the nuclei and include:

- A nucleus not highlighted with a marker;
- Background (stroma);
- A core highlighted with marker;
- Several nuclei.

The case with several nuclei should be considered separately (see subsection 3.4).

The method for determining the intensity and clustering of the color histogram allows you to automatically determine the number of markers and their colors. This saves the precious time of pathomorphologists.

The following classifiers are used for pathomorphological analysis: the support vector method, Bayes classifier, Haar cascade, convolutional neural network [13].

3.4 Methods for the separation of overlapping nuclei

A specific task is the separation of overlapping nuclei. This can be caused by cell division, the camera's viewing angle in the process of shooting the preparation, and also, the location of the nuclei on top of each other in the depth of the examined tissue. The following approaches can be used to separate the fused nuclei and accurately determine their number:

- the method of active contours with the preliminary use of erosion;
- classifiers (convolutional neural network), previously trained in classes that determine the number of cells in the area of a given size;
- watersheds algorithms;
- segmentation algorithms focused on topological features of objects [14].

4. Image dataset

Many images of cell structures and tissues preparations of light and confocal microscopy were collected and labeled (see Table 2).

Material and equipment for shooting images was provided by the Institute of Obstetrics, Gynecology and Reproductology Ott. The shooting of individual classes of images was carried out with a fixed scale of the microscope. Image preparations of various types of tissues with different lighting conditions and marker colors were collected and labelled.

To accelerate nuclei and glands labelling on images, a software was developed that allows a specialist to set a marker on an object of interest in the image. Subsequently, the coordinates of the centers of these markers (x, y), as well as the length (SizeY) and width (SizeX) were recorded in a csv file (see Table 3). Thus, a numerical data of the location and shape of the investigated structures were obtained. The markup was carried out by an employee from the laboratory by a cell biologist at the Ott Research Institute.

Table 2. Collected images

Collected image class	**Number of images**
Confocal microscopy cell preparations	30
Confocal microscopy tissue preparations	92
Lightinh microscopy cell preparations	100

Table 3. A dataset part for the cells

Image name	**sixeX**	**sizeY**	**x**	**y**
1(5).jpg	963	963	278	100
1(5).jpg	963	963	441	201
1(5).jpg	963	963	795	246

1(5).jpg	963	963	911	417
1(5).jpg	963	963	627	475

5. Formulation of the problem

The aim of the work is the research and development of the following algorithms:

- the cell nuclei number estimation with and without researched marker expression on light microscopy image preparations;
- the cell nuclei number estimation with and without researched marker expression on confocal microscopy image preparations;
- highlighting the internal and external glands borders in the confocal microscopy image preparations.

For all algorithms, the following requirements are established:

- work without an operator;
- resistance to changes in the brightness of the preparation;
- resistance to various colors of markers;
- image scale is an input parameter of the algorithms.

6. Suggested algorithms

The algorithms were developed in the PyCharm environment in Python 3.7 using the OpenCV-python 4.0.0.21 library. The source code of developed algorithms is available on github.com [15].

6.1 Counting the number of cell nuclei in confocal microscopy images

- Enter the scaling parameter $expectedPixelsPer100Nm$ – the number of pixels per 100 nanometers;
- Read color image I in RGB format, depth 8 bits per channel;
- Bring the image I to a scale of 1.5 nanometers per pixel;
- Convert I to HSV format, write the V component to the variable V;
- Apply the contrast limited adaptive histogram equalization method with clipLimit = 2 and titleGridSize = 8 on image V;
- Perform erosion on image V with an ellipse core of size 3;
- Calculate mean as the average value of pixels V;
- For each pixel V_{ij}: if $V_{ij} > mean + 20$, assign $V_{ij} = \min(V_{ij} + 100, 255)$ if $V_{ij} < mean - 20$, assign $V_{ij} = \max(V_{ij} - 100, 0)$;
- Apply a median filter with a core of size 5 to the image V;
- Perform threshold binarization of image V with a threshold 127. Write the result to variable B;
- Perform a contour search on image B, leaving contours that do not have nested paths;

Calculate the centers of mass (C_x, C_y) for each contour using formulas (1):

$$C_x = \frac{m_{10}}{m_{00}}; C_y = m_{01}/m_{00}$$
$$m_{p,q} = \sum_{x,y=1..n} x^p y^q \quad (1)$$

Leave only those contours for which $|C_x - C_y| < 1$;

- The number of contours received will be the total number of nuclei.

6.2 Detection of the internal contours of the glands in confocal microscopy images

- Read color image I in RGB format, depth 8 bits per channel;
- Bring the image to a scale of 1.5 nanometers per pixel;
- Convert image I to HSV format, write the V component to the variable V;
- Use the contrast limited adaptive histogram equalization method with parameters $clipLimit$ = 2 and $titleGridSize$ = 8 on image V;
- Perform threshold binarization on V component of HSV with a threshold of 127;
- Perform 27 erosion steps on the image V with the ellipse core of size 3;
- Perform a contour detection on image B, leaving paths that do not have nested paths. Write the result to the contours variable;
- For each contour: calculate the area, count the number of pixels V_{ij} falling into this contour, taking into account the V_{ij} exceeding 15, and write to the variable nonZeroPixelsArea;
    - suppose that the contour is the inner border of the gland if the nonzero pixels of the region exceed the product of the contour of the region 0.4;
- Recognized glands boundaries will be in the contours list an on the output image.

6.3 Counting the number of cell nuclei in light microscopy images

- Enter image I in RGB format and bring it to a scale of 1.5 nanometers per pixel (similar to algorithms A and B);
- Convert I to HSV format, write the V component to the variable V (similar to algorithms A and B);
    - Equalize the histogram of the V component in intensity. To do this,
    - obtain the hist distribution vector of colors (0..255);
    - accumulate the histogram values in $chist$, where $chist[n] = \sum_{i=0..n} hist[i]$ ($hist[i]$);
    - calculate the V_{max} as: $V_{max} = chist[len(chist) - 1] * \frac{(100 - thresh)}{100}$, where $thresh$ is a threshold value equal to 1;
    - replace V pixels with a value greater than V_{max} with a value of V_{max};
- Perform Laplass transforms on image V with sigma equal to 3, 6 and 9, write the results in L_1, L_2, L_3, respectively;
    - calculate the sum of the images L_1, L_2, L_3 by pixels and write to laplaceSum; normalize L_{sum} in the range from 0 to 255;
    - perform Otsu binarization on L_{sum}; write the result to variable B;
- Search for contours in image B, leaving contours that do not have nested paths (similar to algorithm A);

- the result is written to the contours variable;

- Calculate the moments of the contour C_x, C_y according to formulas (1);
  - leave only those contours for which $|C_x - C_y| < 1$;

The task of scaling the image to 1.5 nanometers per pixel was solved by manual measuring the length of the scale bar on the image in pixels. However, we plan to develop automated recognition of the scale bar's length on images

The coefficients of the algorithms were selected by optimizing the criteria by the simplex method and the gradient descent method when testing on the analyzed preparations. The criteria are given in the section 7.

7. Results

7.1 The algorithm for counting the number of cell nuclei in confocal microscopy images

The percent of successful recognition in the researched images was 84%. An example of image recognition is presented in Fig.1.

For the task of counting nuclei, the percentage of successful recognition was presented as the average value for the analyzed images. For each image I, there were the number of detected nuclei N_d and the number of labeled nuclei N_l. The percent of recognition pi for image I was calculated by the formula (2):

$$\boldsymbol{P = \frac{\min(N_d, N_l)}{\max(N_d, N_l)} * 100\%} \quad (2)$$

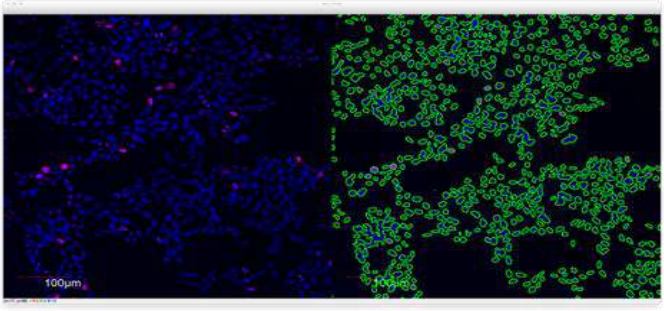


Fig. 1. Highlighting of nuclei on the preparations of confocal microscopy. Initial image (left), image with selected nuclei (right)

7.2 Highlighting of the internal contours of the glands on confocal microscopy images.

The percentage of successful recognition in the researched images was 70%. An example of image recognition is shown in Fig. 2.

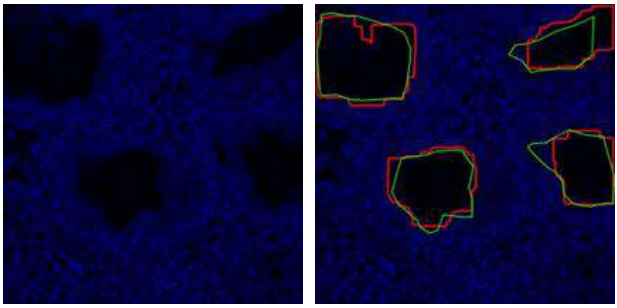


Fig. 2. Highlighting of the internal borders of the glands using confocal microscopy preparations. Initial image (left), image with highlighted glands (right)

For the problem of counting glands, the percentage of successful recognition was given as the average value for the analyzed images. For each image I, there were internal contours of the glands detected by the algorithm – detectedContours and internal contours of the glands marked by the pathomorphologist – labelledContours. The intersection area of the marked and detected contours A_d and the total area of the marked out contours – A_l were calculated. The percent pi of recognition for image I was calculated by the formula (3):

$$\boldsymbol{P = \left|\frac{A_d}{A_l} - 1\right| * 100\%} \quad (3)$$

7.3 The estimation of the number of cell nuclei in light microscopy images

The percentage of successful recognition was 90% in the researched images. An example of image recognition is presented in Fig. 3.

For the task of counting nuclei on light microscopy preparations, the percentage of successful recognition was analyzed by the formula (2) similarly to the percentage of recognition for confocal microscopy.

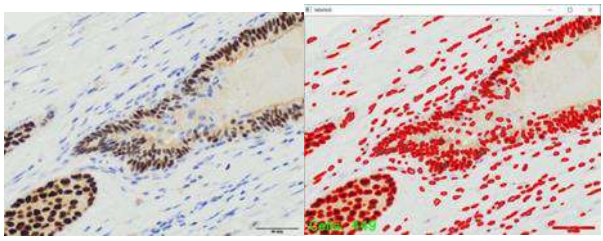


Fig. 3 Highlighting of nuclei on light microscopy preparations. Initial image (left), image with selected nuclei (right)

8. Comparison with existed software

The purpose of last stage of this work was to compare the obtained data with the results of other applications used to evaluate microphotographs.

Since most of the software intended for cytological studies are either expensive or require a long study of the manual and programming languages, it was decided to compare the performance of the created software with FiJi – the ImageJ plugin for evaluating microscopy images, which is the most balanced among its plugins. The initial task was to estimate the number of cells in 30 microphotographs in the jpg format. The comparison results between proposed algorithm and Fiji for confocal imagery task is presented in Table 4.

Table 4. Comparison of created algorithm and FiJi for confocal imagery

No	Labelled	FiJi	% FiJi	Created algorithm	% of created algorithm
1(5)	11	15	73,3	14	78,5
1(6)	11	17	64,7	18	61,1
1(7)	95	109	87,2	94	98,9
1(9)	12	25	48,0	21	57,1
1(13)	360	424	84,9	195	54,2
1(30)	145	261	55,6	165	87,8
1(31)	128	161	79,5	128	100
1(38)	45	59	76,3	57	78,9
1(41)	189	192	98,4	182	96,2
1(42)	123	121	98,4	117	95,1
…					
Result of Fiji				76% +/– 16.1%	
Result of created algorithm				83,8% +/– 11.8%	

For the task of counting nuclei on light microscopy preparations, the percentage of successful recognition was analyzed by the formula (2) similarly to the percentage of recognition for confocal microscopy preparations.

Thus, the developed algorithm exceeds the FiJi accuracy by 7.8%. It should be noted that in order to achieve maximum accuracy in FiJi, the threshold parameter is required and the estimated radius of the object of interest should be introduced as the lower limit, whereas in the written code the determination of the size of objects takes place automatically, which excludes the element of subjectivity from the study and the necessity for preliminary processing of photographs. For example, to evaluate the image, the created algorithm does not require preliminary removal of the scale bar from the image. Also, for the various color markers expression in the nuclei classification task using FiJi, it is necessary to set the Color Threshold value for each type of marker in each photo separately, which not only significantly increases the time of analyze carried out by the pathomorphologist, but also greatly reduces the quality of this analysis, since nuclei having weak expression are most likely will not be included in the corresponding group. Unlike Fiji, the created algorithm equally effectively copes with the task of counting the total number of cores as well as with the task of classifying them.

6. Further research

In the future, it is planned to continue research in this area, improving the reliability of the algorithms, particularly:

use CNN and U-net networks for better segmentation of core images [16, 17];

detect the contours of the scale bar for automatic scaling of the drug;

automatically recognize marker colors (support more than two colors).

References / Список литературы

[1]. Kuznets S.M., Panteleev V.G. The application the Hardware-Software Complex «VideoTesT – Morphology» for the differentiation of tumor cells. Clinic. N.1. 2011, pp. 122-123 (in Russian) / С.М. Кузнец, В.Г. Пантелеев. Использование аппаратно-программного комплекса «ВидеоТесТ – Морфология» для дифференциации опухолевых клеток. Поликлиника, no. 1, 2011 г., стр. 122-123.

[2]. ImageJ software homepage (2020). Available at https://imagej.net.
[3]. VideoTestMorphology software homepage (2020). Available at http://www.digitalimagingsystems.co.uk/pdfs/morpho_en.pdf.
[4]. IMAGE-PRO-Premium software homepage (2020). Available at https://www.mediacy.com/imagepro.
[5]. CellProfiler software homepage (2020). Available at https://cellprofiler.org.
[6]. OrbitImageAnalysis software homepage (2020). Available at https://www.orbit.bio.
[7]. AxioVision 4.8. software homepage (2020). Available at https://www.micro-shop.zeiss.com/en/us/system/software-axiovision+software-products/1007/.
[8]. CellSens software homepage (2020). Available at https://www.olympus-lifescience.com/en/software/cellsens/.
[9]. Berezsky O.N., Melnik G.N. Information technology for the analysis and synthesis of histological images in automated microscopy systems. Control systems and computers, no. 4, 2013, pp. 26-32 (in Russian) / О.Н. Березский, Г.Н. Мельник. Информационная технология анализа и синтеза гистологических изображений в системах автоматизированной микроскопии. Управляющие системы и машины, no. 4, 2013 г., стр. 26-32.
[10]. Carrarelli P., Rocha A. Increased expression of antimullerian hormone and its receptor in endometriosis. Fertility and sterility, vol. 101, no. 5, pp. 1353-1358.
[11]. Drobintseva A.O., Polyakova V.O., Masing D.S., Matyushkin L.B. Confocal microscopy. Role and importance in the study of the reproductive system. Tutorial. SPb., TsOP "Nevsky", 2015, .18 p. (in Russian) / А.О. Дробинцева, В.О. Полякова, Д.С. Мазинг, Матюшкин Л.Б. Конфокальная микроскопия. роль и значение в исследовании репродуктивной системы: учебное пособие. СПб., ЦОП «Невский», 2015 г., 18 стр.
[12]. Chen, S., Zhao, M., Wu, G., Yao, C., Zhang, J. Recent Advances in Morphological Cell Image Analysis. Computational and Mathematical Methods in Medicine, vol. 2012, Article ID 101536, 10 p.
[13]. Gurcan M. N., Boucheron L., Can A., Madabhushi A., Rajpoot N., & Yener B. Histopathological Image Analysis: A Review. IEEE Reviews in Biomedical Engineering, vol. 2, 2009, pp. 147–171.
[14]. Kovrigin A.V. Application of the principles of constructing machine vision systems in the task of analyzing images of cellular structures. Scientific journal of KubSAU, no. 29 (5), 2007, pp. 1-10 (in Russian) / Ковригин А.В. Применение принципов построения систем машинного зрения в задаче анализа изображений клеточных структур. Научный журнал Кубанского государственного аграрного университета, no. 29(5), 2007 г., стр. 1-10.
[15]. Xue Y., Ray N. (Cell Detection with Deep Convolutional Neural Network and Compressed Sensing. arXiv:1708.03307, 2007.
[16]. Falk, T., Mai, D., Bensch, R. et al. U-Net: deep learning for cell counting, detection, and morphometry. Nature Methods, vol. 16, 2019, pp. 67–70 (2019).
[17]. Sergreev D.I., Drobintseva A.O. et al. Biomedicine diagnostic repository. Source code of this paper. Available at https://github.com/densvr/biomedicine-diagnostic.

Information about authors / Информация об авторах

Daniel Igorevich SERGEEV – PhD student of the Institute of Computer Science and Technology. Scientific interests: computer vision, machine learning, development in the Android environment.

Даниэл Игоревич СЕРГЕЕВ – аспирант Института компьютерных наук и технологий. Научные интересы: компьютерное зрение, машинное обучение, разработки в среде Андроид.

Alexander Evgenievich ANDREEV – graduate of the magistracy of SPbPU, researcher. Research interests: biomedical diagnostics, computer vision, machine learning.

Александр Евгеньевич АНДРЕЕВ – выпускник магистратуры СПбПУ, исследователь. Научные интересы: биомедицинская диагностика, компьютерное зрение, машинное обучение.

Anna Olegovna DROBINTSEVA – Associate Professor, Candidate of Biological Sciences, Associate Professor of the Department of Medical Biology. Research interests: biomedical diagnostics, molecular markers, peptide hormones, infertility, microscopy.

Анна Олеговна ДРОБИНЦЕВА – доцент, кандидат биологических наук, доцент кафедры медицинской биологии. Научные интересы: биомедицинская диагностика, молекулярные маркеры, пептидные гормоны, бесплодие, микроскопия.

Nikola KUKAVITSA is a graduate student of the Institute of Computer Science and Technology. Scientific interests: computer vision, machine learning, python

Никола КУКАВИЦА – студент магистратуры Института компьютерных наук и технологий. Научные интересы: компьютерное зрение, машинное обучение, питон.

Slobodanka CENEVSKA – graduate student at the Institute of Computer Science and Technology. Research interests: computer vision, machine learning, python.

Слободанка ЦЕНЕВСКА – студентка магистратуры Института компьютерных наук и технологий. Научные интересы: компьютерное зрение, машинное обучение, питон.

Pavel Dmitrievich DROBINTSEV – Associate Professor, Candidate of Technical Sciences, Director of the Higher School of Software Engineering at the Institute of Computer Science and Technology. Research interests: computer vision, machine learning, python.

Павел Дмитриевич ДРОБИНЦЕВ – доцент, кандидат технических наук, директор Высшей школы программной инженерии Института компьютерных наук и технологий. Научные интересы: компьютерное зрение, машинное обучение, питон.

DOI: 10.15514/ISPRAS-2020-32(3)-12

Анализ загруженности трафика на главных улицах электронного города с применением индекса перегрузки и искусственной нейронной сети (на примере города Хамедан)

[1]*М.М. Ширмохаммади, ORCID: 0000-0001-8858-3770 <mmshirmohammadi@iauh.ac.ir>*
[2]*М. Эсмаилпур, ORCID: 0000-0002-2475-518X <esmaeilpour@iauh.ac.ir>*
[1]*Аракский филиал Исламского университета Азад,*
Иран, Арак, 3-й км дороги Хомейна, площадь Имама Хомейни
[2]*Хамеданский филиал Исламского университета Азад,*
Иран, Хамедан, бул. Мусиванд, Мадани Таун

Аннотация. Заторы на дорогах являются серьезной проблемой для электронных городов, и для решения этой проблемы необходимо анализировать пробки в городской дорожной сети. В этой статье изучается показатель эффективности транспортных средств для оценки условий дорожных сетей. В нашем исследовании исследуется плотность трафика главной дорожной сети города Хамедан на основе данных о скорости, собранных системой управления движением Хамедана. На основе этого анализа были определены индекс трафика и пиковые часы трафика. Кроме того, с использованием нейронной сети и генетического алгоритма была определена предсказуемая связь между скоростью транспортных средств и загруженностью трафика. В работе использовались данные Центра управления движением Хамедана о скорости движения транспортных средств в густонаселенных районах.

Ключевые слова: пробки на дорогах; эффективность скорости; городские дорожные сети; управление и контроль трафика; нейронная сеть; генетический алгоритм

Для цитирования: Ширмохаммади М.М., Эсмаилпур М. Анализ загруженности трафика на главных улицах электронного города с применением индекса перегрузки и искусственной нейронной сети (на примере города Хамедан). Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 131-146. DOI: 10.15514/ISPRAS–2020–32(3)–12.

Analysis of Traffic Congestion in Main Streets of Electronic city using Traffic Congestion Index and Artificial Neural Network (Case Study: Hamedan City)

[1]*M.M. Shirmohammadi, ORCID: 0000-0001-8858-3770 <mmshirmohammadi@iauh.ac.ir>*
[2]*M. Esmaeilpour, ORCID: 0000-0002-2475-518X <esmaeilpour@iauh.ac.ir>*
[1]*Islamic Azad University Arak Branch,*
3rd km of Khomein road, Imam Khomeini Square, Arak, Iran
[2]*Islamic Azad University Hamedan Branch,*
Mousivand Blvd., Madani Town, Hamedan, Iran

Abstract. Smart cities are a kind of umbrellas of different technologies for responding to the problem of increasing urban population. The priority of intelligent electronic cities is a strategy to collecting information about the city and its smart use to improve the provided services to citizens or to create new services. These smart cities have weather forecast, urban monitoring, pollution monitoring and various applications. Traffic is

a major challenge for electronic cities and coping with it requires analyzing traffic congestion in the city road network. The data transmission with wireless signals in smart cities is one of the challenges because construction of high buildings and barriers reduces the power and quality of the signal. Widespread use of wireless signals and equipment may lead to interference and reduce service quality. Therefore, in order to solve the traffic problem, it is necessary to achieve traffic congestion levels by collecting information, especially with wireless signals so that it can be programmed to control and manage traffic. In this paper, the performance index of vehicle speed was estimated to evaluate the conditions of road networks. This study analyzes the traffic density for the main network of Hamedan communication routes based on the collected data of Speed performance of Hamedan traffic control system. According to this analysis, the congestion index and traffic peak hours were determined. Also the relationship between vehicle speed and traffic congestion was predicted by neural network and the genetic algorithm. In this study areas of traffic were identified using Hamedan Traffic Control Center according with the speed of vehicles.

Keywords: Traffic Congestion; Speed Performance; Urban Road Network; Traffic Management and Control

For citation: Shirmohammadi M.M., Esmaeilpour M. The Traffic Congestion Analysis using Traffic Congestion Index and Artificial Neural Network in Main Streets of Electronic city (Case Study: Hamedan City). Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020. pp. 131-146 (in Russian). DOI: 10.15514/ISPRAS–2020–32(3)–12

1. Введение

Основой электронных интеллектуальных городов являются различные технологии, помогающие решить проблему роста городского населения. Приоритетным направлением является сбор информации о городе и его интеллектуальном использовании для улучшения услуг, предоставляемых гражданам, или для создания новых услуг [1]. В интеллектуальных городах имеются приложения для прогноза погоды, мониторинга градостроительства, мониторинга загрязнений и других разнообразных применений [2].

Проблемой больших городов являются пробки, и одним из решений этой проблемы является повышение уровня интеллектуальности города. Владение точной информацией о ситуации в городе может помочь принять важные решения в городском управлении. В масштабе города могут использоваться различные датчики, объединенные в беспроводную сенсорную сеть и собирающие ценную информацию [3].

Однако беспроводная передача данных в интеллектуальных городах затруднительна, потому что наличие высоких зданий приводит к снижению качества сигналов [4]. Широкое использование беспроводного оборудования может привести к помехам и снижению качества обслуживания [5]. Поэтому, чтобы решить проблему перегрузки трафика, требуется собирать информацию (с использованием беспроводных сетей), чтобы на ее основе обеспечить контроль трафика и управление им.

В настоящее время нет фиксированных и стабильных оценочных инструментов для оценки состояния трафика; фактически, в разных регионах используются различные меры и оценки, основанные на конкретных приложениях и потребностях. Однако можно вычислить среднее время пиковой нагрузки трафика путем определения индекса оценки загруженности.

Во втором разделе статьи приведен обзор работ, связанных с электронными городами. В третьем разделе обсуждается предлагаемый метод для расчета индексов. В четвертом разделе описывается применение этого метода в условиях города Хамедан, и в последнем, пятом разделе представлено заключение.

2. Состояние области

Интеллектуальные города разного масштаба были созданы во многих крупных городах мира различными способами для конкретных применений, включая управление трафиком. Один из таких проектов был реализован в Сингапуре. Этот проект основан на облачных вычислениях, он фокусируется на интеллектуальных транспортных задачах, имитирует и

оценивает структуру потока данных, характеризующих трафик. Для оценки ситуации с дорожным движением в этом проекте были выделены параметры дорожных сетей, были изучены сценарии трафика и разработаны алгоритмы извлечения данных [6].

В Азии одним из самых развитых интеллектуальных городов является Сонгдо (Южная Корея). Это полностью цифровой город, в котором все информационные системы взаимосвязаны, и практически все объекты привязаны к информационной системе [7]. Этот проект предназначен для продвижения интересов частного бизнеса, не обращая внимания на потребности сообщества рядовых граждан [8].

В одном из успешных проектов по созданию интеллектуального города в г. Сантандер в Испании использовались беспроводные датчики в среде интернета для измерения концентрации угарного газа, интенсивности света, шума, температуры и движения транспортных средств [9].

Барселона [10] является ещё одним примером успешного интеллектуального города, который известен как самый умный город в мире. В Барселоне имеется мощная платформа с комплексной инфраструктурой. Эта технология обеспечивает связность элементов города и позволяет им легко взаимодействовать друг с другом, а также управлять ими с помощью электронных устройств. Интеллектуальная модель Барселоны предусматривает 12 регионов с экологическими проектами, ICT, мобильностью, водой, энергией, управлением отходами, природой, территорией и сооружениями, открытым пространством и услугами.

Цзя (Shunping Jia) и др. получили всестороннюю оценку моделей городского трафика на основе средней скорости движения транспортных средств с учетом характеристик простоя в движении и пропускной способности дорожной сети [11]. Чжу (Fuling Zhu) исследовал систему оценки индекса загруженности городского транспорта и распределение скорости автотранспортного средства с использованием смешанной модели Гаусса для характеристик плотности [12].

Кирога (Cesar A. Quiroga) использовал время нахождения в пути для измерения эффективности транспортной сети. В своей работе он обсуждает методы сбора данных о времени поездки и скорости [13].

Роберт (R. Robert) и др. обсудили понятия времени нахождения в пути и длительности поездки и изучили влияние различных показателей на качество загруженности, а также предложили метод классификации перегрузок на основе времени поездки с точки зрения путешественников. В их работе также отмечается, что сложность и динамический характер трафика являются проблемой, которую трудно решить с использованием только одного оценочного индекса. В результате различных исследований трафик стали оценивать на основе нескольких показателей [14].

Бартини (Robert L. Bertini) и др. использовали время нахождения в пути транспортного средства, данные о скорости движения в Портленде (шт. Орегон, США) для оценки условий дорожного движения [15,16].

Коифман (Benjamin Coifman) и др. использовали средства автоматического определения местоположения автомобилей для измерения времени движения и средней скорости автомобилей на автомагистралях, а также определения существующей дорожно-транспортной обстановки [17].

Дуан (Houli Duan) и др. применили объем и плотность трафика для анализа и наглядного представления трафика [18].

Турочи (Rod Turochy) и др. вычисляют индекс изменчивости на основе многофакторной статистики, используя большую архивную коллекцию данных о трафике. Индекс изменчивости позволяет определять время суток и дни недели, в которые наблюдается высокая степень изменчивости дорожно-транспортной обстановки [19].

Ван (Y. Wang) и др. оценивают дорожно-транспортную обстановку на основе значений переменных трафика, таких как средняя скорость и плотность трафика. Недостатками подхода являются сложность вычислений, трудности в сборе требуемых наборов данных и т.д. [20].

Влахоянни (Eleni I. Vlahogianni) и др. предложили способ объединения временных и пространственных характеристик потока трафика и его оптимизации с помощью генетических алгоритмов, позволяющий лучше прогнозировать потоки трафика по сравнению с другими методами [21].

Инь (Hongbin Yin) и др. использовали нечеткую нейронную модель для прогнозирования потока трафика. Используя нечеткий метод, они разбивают входные данные на несколько кластеров, а затем на основе нейронной сети определяют взаимосвязи между входными и выходными данными. Это увеличивает прогнозирующую способность модели за счет адаптивного регулирования ее коэффициентов в соответствии с текущей дорожно-транспортной обстановкой [22].

Люй (Yisheng Lv) и др. предложили новый метод прогнозирования потока трафика на основе глубокого обучения. В этом методе учитываются корреляции пространственных и временных показателей [23].

Смит (Brian Lee Smith) и др. занимались краткосрочным прогнозированием потока трафика с использованием генетического подхода и показали, что для этого хорошо подходят модели ближайших соседей [24].

Чен (Dawei Chen) и др. предложили комбинацию нейронной сети и пчелиного алгоритма для среды больших данных, которая использовалась в условиях плотного трафика для прогнозирования его потоков и обеспечивала большую точность, чем предыдущие методы [25].

Ма (Xiaolei Ma) и др. предложили использовать для краткосрочного прогнозирования трафика нейронную сеть с кратковременной памятью, которая уменьшает ошибку прогнозирования и позволяет прогнозировать скорость движения [26].

Абдулхай (Baher Abdulhai) и др. разработали метод краткосрочного прогнозирования транспортных потоков, основанный на пространственной информации, и обнаружили, что если исключить пространственную информацию, то ошибка прогнозирования удваивается [27].

Массобрио (Massobrio) и др. Представили открытую транспортную систему, использующую исторические данные автобусного движения для прогнозирования мобильности пассажиров на основе данных о продаже билетов с помощью смарткарт [28].

Аларкон-Акино (Vicente Alarcon-Aquino) и др. предложили многофункциональную нейронную сеть, в которой используются методы теории вейвлетов и которая функционирует лучше некоторых линейных прогнозных систем [29].

Чен (Yuehui Chen) и др. использовали гибкую модель нейронного дерева для мелкомасштабного прогнозирования трафика, которая обеспечивает лучшую точность прогнозирования и меньшую погрешность, чем предыдущие методы [30].

Фаббиани (Fabbiani) и др.. строили матрицу исходных и конечных пунктов движения на основе анализа продаж билетов и сведений о локализации автобусов [31].

Янг (Hong-jun Yang) и др. предложили кластерную модель прогнозирования с использованием генетического алгоритма. Модель позволила обеспечить точность краткосрочного прогнозирования потока трафика, более высокую, чем у предыдущих методов. [32].

Лу (Baichuan Lu) и др. для прогнозирования потоков трафика использовали реальные данные трафика, искусственную нейронную сеть и волновые преобразования. Метод позволяет повысить точность предсказаний за счет большего объема вычислений [33].

3. Предлагаемый метод

Согласно предыдущим исследованиям, скорость транспортного средства является показателем оценки плотности движения. В нашем исследовании индекс скорости транспортного средства определяется на основе средней скорости движения, максимальной разрешенной скорости и классификационного индекса загруженности дорог.

3.1 Индекс эффективной скорости

Скорость транспортного средства является важным параметром для измерения плотности трафика. Скорость не может превышать 100 км/час, и индекс скорости (со значениями от 0 до 100) представляет собой частное от деления скорости транспортного средства на максимальную разрешенную скорость. В нашем исследовании для оценки дорожно-транспортной обстановки используется индекс эффективной скорости. Для классификации городской дорожно-транспортной обстановки были выбраны четыре пороговых значения: 25, 50, 75, 100. Чем меньше значение индекса эффективной скорости, тем медленнее движется транспорт и тем вероятнее ситуация затора. Основываясь на таких оценках анализ загруженности городской дорожной сети можно вести на основе индекса плотности дорожного сегмента и его измеренных значений. Например, если значение индекса эффективной скорости находится в диапазоне от 0 до 25, среднюю скорость надо считать очень низкой, а загруженность дороги очень высокой. Диапазон от измеренных значений 25-50 показывает среднюю загрузку и невысокую среднюю скорость. В диапазоне 50-75 скорость уже выше, а диапазон 75-100 показывает высокую скорость и хорошие условия для движения транспорта.

Индекс скорости вычисляется по формуле (1). В этой формуле $\boldsymbol{V}$ – средняя скорость движения, $\boldsymbol{V_{max}}$ – максимально разрешенная скорость на дорогах и $\boldsymbol{R_v}$ индекс эффективной скорости:

$$R_v = \frac{V}{V_{max}} \times 100 \tag{1}$$

3.2 Индекс плотности загрузки дорожного сегмента

Для анализа заторов на дорогах используется индекс плотности загрузки дорожного сегмента городских дорожных сетей. Для расчета индекса плотности загрузки сегмента дороги наблюдение велось за трафиком на середине этого сегмента. Индекс плотности дорожного сегмента вычисляется по формулам (2) и (3). Этот индекс находится между 0 и 1, и чем меньше число, тем больше затруднения в трафике. В этих уравнениях $\boldsymbol{R_i}$ представляет собой индекс плотности загрузки дорожного сегмента, $\boldsymbol{R_v}$ – индекс эффективной скорости, $\boldsymbol{R_{NC}}$ – доля времени пребывания дорожного сегмента в незагруженном состоянии, $\boldsymbol{t_{NC}}$ – время незагруженного состояния (в минутах) и $\boldsymbol{t_T}$ – период наблюдения (в минутах).

$$R_i = \frac{\overline{R_v}}{100} \times R_{NC} \tag{2}$$

$$R_{NC} = \frac{t_{NC}}{t_T} \tag{3}$$

3.3. Индекс плотности загрузки дороги

Дорожная сеть состоит из участков дороги, индекс плотности загрузки сегмента дороги вычисляется по формуле (4). Значение индекса плотности дорожной сети находится в пределах от 0 до 1, и чем меньше число, тем больше плотность загрузки дорожной сети. В этой формуле $\boldsymbol{R}$ – это индекс загрузки дорожной сети, а $\boldsymbol{L_i}$ – длины отдельных сегментов дорожной сети.

$$R = \frac{\sum_i R_i L_i}{\sum_i L_i} \quad (4)$$

4. Опыт проведенных исследований

Авторами были рассмотрены характеристики транспортного движения города Хамедан. В течение почти 10 лет в этом городе проводились активные мероприятия для создания стабильной электронной городской среды. Основная транспортная сеть города Хамедан состоит из двух колец, одного полукольца и шести соединительных линий между кольцами. Авторами была проанализирована информация, полученная на основе данных центра управления движением Хамедана. На рис. 1 показана карта основных дорог города Хамедан, видны кольца и соединительные линии между ними.

Рис. 1. Схема главных дорог города Хамедана и городской центр управления движением
Fig. 1. Scheme of the main roads of the city of Hamedan and the city traffic control center

Результат исследования показывает, что наиболее часто встречающееся значение индекса эффективности скорости, вычисленного по всему массиву зафиксированных в Хамедане скоростей, составляет 65, что превышает 50%. С ростом скорости до значения индекса 60 загруженность дорожной сети растет медленно, а затем темп ее роста возрастает. Средний индекс эффективной скорости на главных дорогах Хамедана, рассчитанный для разного времени суток, приведен в табл. 1.

В таблице 1 строка ***WD*** представляет рабочие дни недели, а строка ***WE*** – выходные и праздничные дни. Строка ***D*** содержит разницу между этими днями. Из таблицы видно, что самые большие различия в скоростях автомобилей зафиксированы в 8, 13 и в 15 часов, и в рабочие дни в эти часы транспорт движется с меньшей скоростью из-за возрастающего трафика. Средняя скорость автомобилей в рабочий день составляет 67,08, средняя скорость в конце недели достигает 69,95.

Табл. 1. Сравнение средней скорости на главных дорогах Хамедана в рабочие (WD) и выходные (WE) дни
Table 1. Comparison of average speed on the main roads of Hamedan on working days (WD) and weekends (WE)

Время суток (час)	0	1	2	3	4	5	6	7	8	9		10	11
Рабочие дни (WD)	72	78	82	90	95	85	81	75	55	60		65	62
Выходные дни (WE)	75	75	80	87	93	82	79	75	72	70		70	65
Разница (D)	3	-3	-2	-3	-2	-3	-2	0	17	10		5	2
Время суток (час)	12	13	14	15	16	17	18	19	20	21		22	23
Рабочие дни (WD)	58	50	55	57	60	65	63	56	59	60		62	65
Выходные дни (WE)	64	63	65	70	65	65	60	55	60	62		63	64
Разница (D)	6	13	10	13	5	0	-3	-1	1	2		1	-1

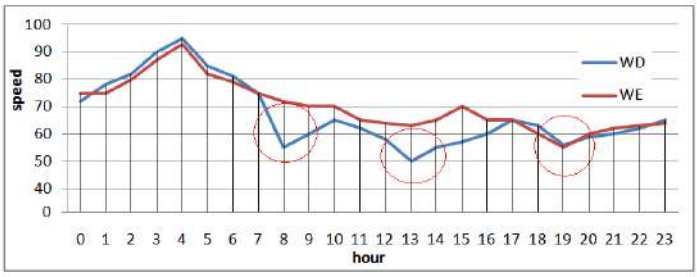


Рис. 2. Почасовой индекс эффективности скорости в среднем за один день
Fig. 2. Hourly average speed efficiency index for one day

Рис. 2 показывает почасовые индексы эффективной скорости в рабочие и выходные дни. Пиковые часы в будние дни обычно возникают утром с 07:00 до 09:00, днем с 12:30 до 14:30, а вечером с 17:00 до 19:00. Самые высокие нагрузки наблюдаются утром, днем и вечером в 8:00, 13:00 и 18:00. Время вечерней пиковой нагрузки на дороги оказывается разным в выходные и будние дни, но утром и днем в будние дни трафик увеличивается больше, чем в выходные. Кроме того, психологически важные часы трафика приходятся на раннее утро, и при этом нет большой разницы между выходными и будними днями.

4.1 Оценка перегрузки сети

На рис. 3 показаны три кривые, демонстрирующие загруженность дорожной сети Хамедана в будние дни. Изменения состояния сети в часы пик утром, днем и вечером, соответственно, показаны с помощью расчета индекса эффективной скорости на дороге каждые 15 минут. Утром, днем и вечером на рисунке 3 (a, b и c) состояние дорожной сети ухудшается, но совсем рано утром ситуация спокойнее.

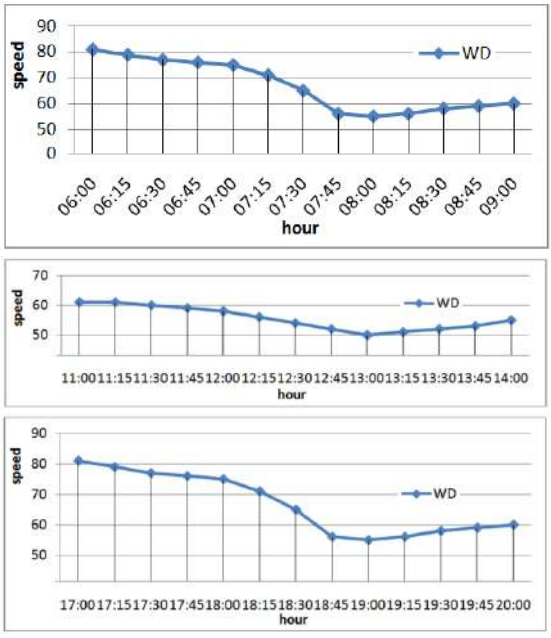


Рис. 3. Показатели индекса эффективной скорости в часы пик
Fig. 3. Indicators of the index of effective speed during rush hours

4.2 Прогнозирование загруженности нейронной сетью

Авторы провели исследование изменений скоростей транспортных средств и плотность движения во времени, при этом выявлялась связь средней суточной плотности трафика со снижением скорости движения. Анализ данных о скорости автомобилей проводился с использованием статистического коэффициента корреляции Пирсона, уравнений линейной регрессии, полиномиальной регрессии (полулинейный метод) [34-35] и искусственных нейронных сетей [36-37]. Для точного прогнозирования плотности трафика использовались искусственная нейронная сеть и генетические алгоритмы.

Первоначально данные, используемые в искусственной нейронной сети, были разделены на две отдельные части, причем половина данных предназначалась для обучения сети, а вторая половина данных использовалась для тестирования. В качестве функции стимуляции в методе пост-распространения используется функция Танксона (Tanxon). Эта функция преобразует интервал значений каждого нейрона в интервал, ограниченный значениями -1 и 1. Значения из этого сжатого интервала рассматриваются как коэффициент корреляции. Коэффициент корреляции показывает соотношение между выходом сети и фактическим значением исследуемого параметра. Значение 1, означает точное следование выхода сети за изменением фактического параметра, а -1 означает, что изменения выхода сети и фактического параметра происходят в противофазе. В качестве входного параметра рассматривалась скорость транспортного средства, а выход сети трактовался как плотность трафика. Чтобы уменьшить ошибку прогнозирования, при расчетах параметров сети применялся метод проб и ошибок. Для обучения сети использовалась модель многослойного перцептрона. Для обучения сети использовался один скрытый слой и Дельта-правило с коэффициентом обучения 0.1, контролировавшим степень соответствия весов связей между нейронами.

Количество нейронов скрытого слоя для сетевого обучения было выбрано равным 4 и момент, который является одним из основных параметров обучения и определяет воздействие исходных значений весов связей на новые значения этих весов, равен 0.7.

При анализе регрессионного и полулинейного методов полиномиальный коэффициент корреляции между плотностью трафика и минимальной скоростью транспортного средства составил 0.88% в течение второй половины 2017 года, что является значимым на уровне $\alpha = 0.01$, а коэффициент определения рассчитан как 0.77%. Этот анализ показывает, что между средней дневной плотностью трафика и минимальной скоростью движения существует обратная зависимость.

При использовании искусственной нейронной сети результаты, полученные в сети, обученной на регулярно поставляемых примерах, оказываются более хорошими, чем в том случае, когда выбор обучающих примеров осуществляется случайным образом. В реальной жизни метод обратного распространения ошибки (BP) часто работает очень медленно. Для преодоления этой проблемы используется генетический алгоритм выбора лучших первичных весов связей. Ускорения получения результатов можно добиться использованием нейронной сети и ее сочетанием с генетическим алгоритмом.

Сравнение среднесуточной плотности трафика с его прогнозируемым уровнем, полученным на основе случайных данных с использованием генетического алгоритма, показывает, что минимальная среднеквадратичная ошибка (MMSE) в этом режиме равна 0.01, при этом коэффициент корреляции между среднесуточной плотностью трафика и выданным сетью со случайными обучающими данными прогнозом и генетическим алгоритмом составляет 94%, а коэффициент детерминации составляет 0.90.

Сравнение методов линейной и полиномиальной регрессии с использованием искусственной нейронной сети показало, что коэффициент детерминации зависимости средней плотности

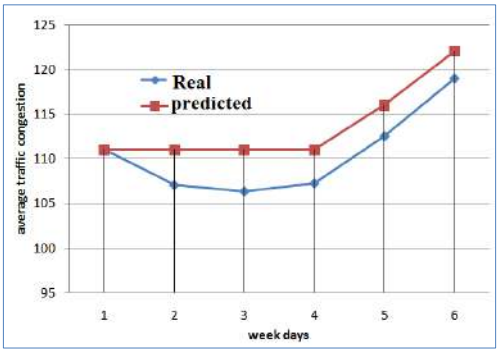
трафика с минимальной скоростью транспортного средства в линейной регрессии равен 0.77, в полиномиальной регрессии равен 0.90, а в нейронной сети равен 0.94 (таблица 2).

Табл. 2. Сравнение коэффициентов линейной и полиномиальной регрессии с нейронной сетью
Table 2. Comparison of the coefficients of linear and polynomial regression with a neural network

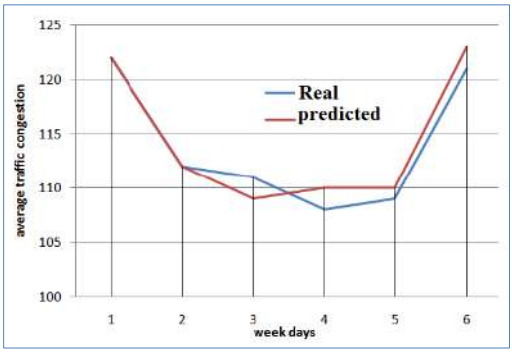
	Нейронная сеть	Полиномиальная регрессия	Линейная регрессия
Средний коэффициент детерминации	0.94	0.90	0.77

Согласно полученным результатам можно сказать, что нейронная сеть хорошо предсказывает взаимосвязь между средней дневной плотностью трафика и скоростью транспортных средств.

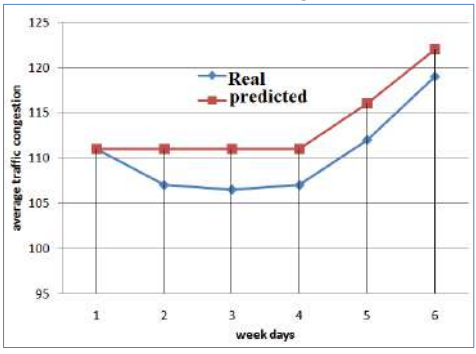
На рис. 4 показано сравнение среднесуточного значения фактической плотности трафика с прогнозируемым числом, полученным на случайных данных совместно с генетическим алгоритмом. Там же показано сравнение среднесуточного фактического количества плотности трафика с прогнозируемым числом, полученным на упорядоченных данных и без генетического алгоритма, а также средние ежедневные данные плотности трафика в упорядоченных данных вместе с генетическим алгоритмом.



a) случайные данные и генетический алгоритм
a) random data and genetic algorithm



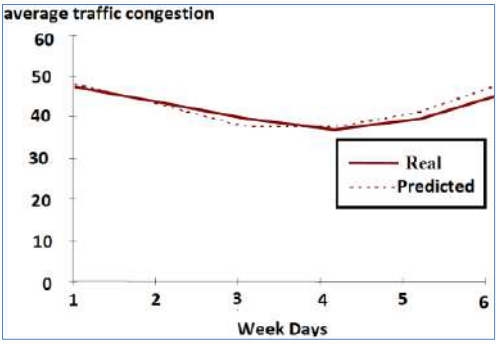
b) регулярные данные без применения генетического алгоритма
b) regular data without the use of a genetic algorithm



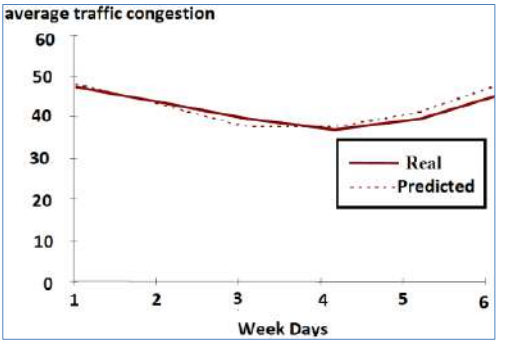
c) регулярные данные с применением генетического алгоритма
c) regular data using the genetic algorithm

Рис. 4. Сравнение фактической и прогнозируемой плотности трафика
Fig. 4. Comparison of actual and forecasted traffic density

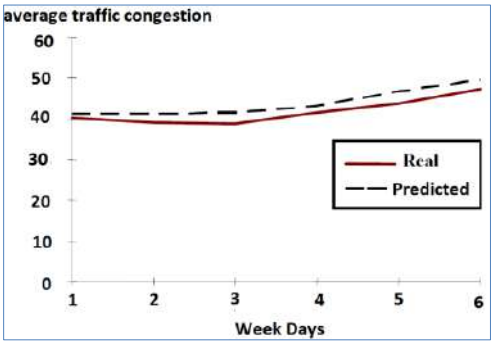
На рис. 5 показаны результаты последующих исследований, выполненных для сравнения данных о фактическом росте загрузки дорог, полученных при наблюдениях в Хамедане, с прогнозами загрузки дорог, рассчитанными на основе использования нейронной сети.



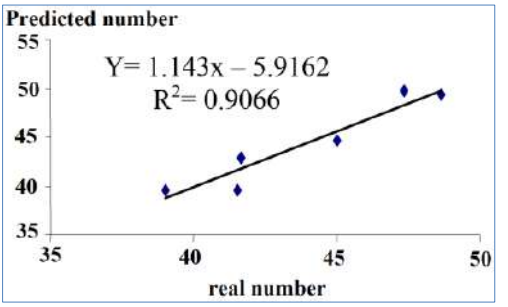
a) случайные данные и генетический алгоритм
a) random data and genetic algorithm



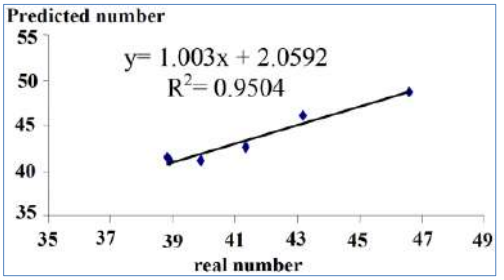
b) регулярные данные без применения генетического алгоритма
b) regular data without the use of a genetic algorithm



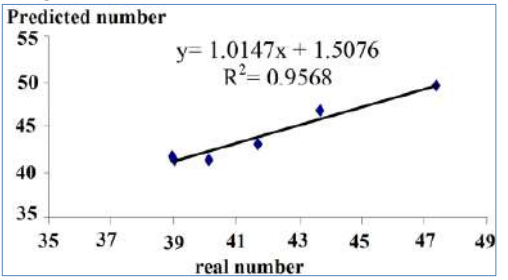
c) регулярные данные с применением генетического алгоритма
c) regular data using the genetic algorithm



d) Средняя корреляция, вычисленная на случайных данных с использованием нейронной сети и генетического алгоритма
d) Average correlation calculated on random data using a neural network and a genetic algorithm



e) Средняя корреляция, вычисленная на регулярных данных с использованием нейронной сети
e) Average correlation calculated on regular data using a neural network



f) Средняя корреляция, вычисленная на регулярных данных с использованием нейронной сети и генетического алгоритма
f) Average correlation calculated on regular data using neural network and genetic algorithm

Рис. 5. Среднесуточное сравнение фактической и прогнозируемой плотности трафика
Fig. 5. Average daily comparison of actual and forecasted traffic density

В качестве практического результата проведенного исследования была получена возможность достаточно точного определения географического положения транспортного затора на карте города по выявляемым показателям снижения средней скорости потока.

Выявление узких мест помогает ставить задачу изменения или, по крайней мере, планирования изменений геометрии транспортных путей для балансировки нагрузки на всех альтернативных путях и последующего снижения степени перегруженности сети.

На рис. 6 показаны участки плотных транспортных пробок в городе Хамедан. На этом рисунке видно, что неравномерности в росте города Хамедан приводят к затруднениям движения в некоторых частях города в часы пик. С другой стороны, в другой части города загруженности трафика не наблюдается, и движение продолжается без особых затруднений. Приведенная иллюстрация показывает, что в часы пиковой загруженности трафика те точки на карте городских районов, где имеются затруднения в движении транспорта, должны быть исправлены.

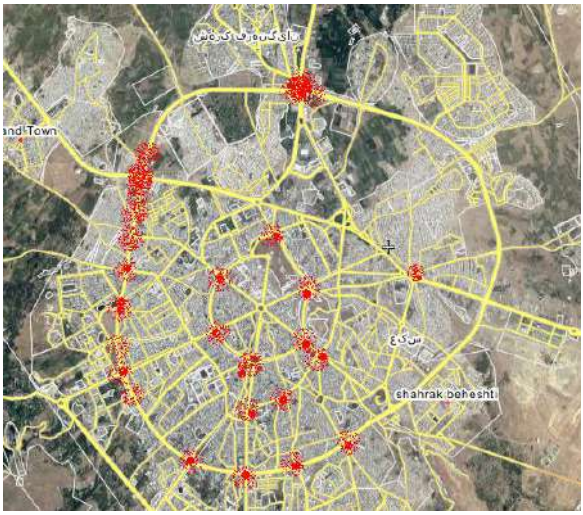


Рис. 6. Отображение мест плотного движения в городе Хамедан в часы пик
Fig. 6. Display of heavy traffic in Hamedan during rush hours

Авторы тестировали возможность использования специализированного программного инструмента, разработанного компанией IBM (IBM Traffic Prediction Tool) для исследования условий движения транспорта в Сингапуре. Этот инструментарий позволяет также делать прогнозы относительно времени возвращения трафика к нормальному уровню, что достигается использованием видеокамер, системы позиционирования GPS, системы управления такси и датчиков, установленных на улицах. Такой подход оказывается значительно более дорогостоящим, чем метод, предложенный в настоящей статье, который не требует дорого оборудования и существенного накопления больших массивов информации [38].

Авторами была рассмотрена модель, предложенная компанией Яндекс (Yandex), которая выдает рекомендации по управлению трафиком на основе собираемой информации. При выработке этих рекомендаций учитываются географические координаты, порядок и скорость движения транспорта. Ошибки в отслеживании положения транспортных средств на дороге исправляются с помощью сигналов от глобальных систем позиционирования (например, GPS). Система позволяет находить правильный маршрут, базируясь на собственном методе маршрутизации, и хорошо подходит для крупных пространств. Проект Яндекса зависит от глобального позиционирования, в то время, как метод, представленный в этой статье, такой зависимости не имеет [39].

Для повышения точности проведения исследований в Хамедане был выбран район с напряженным потоком движения. Было проведено моделирование реального перекрестка – на южной стороне площади Джахад в городе Хамедан. Этот перекресток окружен жилыми и коммерческими кварталами, ключевыми зданиями, он характеризуется большим количеством пешеходов и транспортных средств. В данном случае рассматривались только потоки транспорта в определенные рабочие дни, при этом изучались различия в характере движения. Использовались данные о потоках движения с 6 до 8 марта и с 23 до 25 мая (фиксация данных в каждом интервале осуществлялась каждые 15 минут). Были получены и изучены 288 наборов данных по каждому временному интервалу, то есть всего 576 наборов. Например, для прогнозирования транспортных потоков использовались данные, полученные с нуля часов 23-го мая до 24 часов 25-го мая.

Сравнение методов оценки, выполнявшееся авторами настоящей работы, базируется на концепции анализа временных рядов ARIMA (авторегрессивное интегрированное скользящее среднее), используемой в качестве модели нестационарного временного ряда, формировавшегося с 1979 по 2018 год с некоторыми изменениями, вносившимися для более полного использования преимуществ конкретных приложений. Тань (Tang) в 2018 году, используя ту же концепцию, рассматривал транспортный поток как нестационарную случайную последовательность [40]. Другим возможным методом анализа является разрабатывавшийся с 2002 по 2015 год метод WNN (вейвлет-нейронная сеть), то есть нелинейная волновая нейронная сеть. Этот метод основан на использовании вейвлета в качестве функции активации для скрытых узлов и весов входных нейронов [41].

Циньчжун Хоу (Qinzhong Hou) и его коллеги также предложили комбинацию вышеуказанных методов, названную ими гибридной моделью ARIMA & WNN, в которой веса назначаются на основе нечетких вычислений [42]. Для этой модели, представленной ее авторами в 2019 году, были по формулам (5) в процентах рассчитаны фактическая относительная погрешность выхода RPE (relative pointing error), средняя процентная абсолютная ошибка MAPE (mean percentage absolute error) и корень среднеквадратичной погрешности RMSE (root mean square error). В приведенных формулах $\boldsymbol{u_t}$ – наблюдаемый трафик, $\boldsymbol{Q_t}$ –результат прогнозирования, $\boldsymbol{l}$ – количество сделанных наблюдений [42].

$$MAPE = \frac{1}{l}\sum_{t=1}^{l}\left|\frac{u_t - Q_t}{u_t}\right| \times 100,$$

$$RPE = \left|\frac{u_t - Q_t}{u_t}\right| \times 100,$$

$$RMSE = \sqrt{\left(\frac{1}{l} \times \sum_{t=1}^{l}\left(\frac{u_t - Q_t}{u_t}\right)^2\right)} \times 100. \qquad (5)$$

Табл. 3, содержащая данные о сравнении различных моделей оценки результатов моделирования, показывает, что предлагаемый авторами метод намного лучше, чем предыдущие методы. В этом методе с учетом различий в трафике в разные месяцы были рассмотрены два сценария трафика в марте месяце и трафик с фиксированной скоростью в мае. Предложенный метод (сам по себе и в сочетании с вейвлет-нейронной сетью) был также сравнен с ранее использовавшимися методами.

Табл. 3. Сравнение предлагаемого метода оценки результата с ранее использовавшимися методами по данным, полученным в марте и мае соответственно
Table 3. Comparison of the proposed method for evaluating the result with previously used methods according to the data obtained in March and May, respectively

index	ARIMA	WNN	HYBRID ARIMA&WNN	PROPOSED1	PROPOSED 2 &WNN
RPE	15.64	13.49	4.07	3.97	3.12
MAPE	9.91	7.65	5.98	5.42	4.83
RMSE	8.49	7.31	6.12	5.72	5.01
RUNNINGTIME	14.33	3.59	17.36	12.41	10.24

index	ARIMA	WNN	HYBRID ARIMA&WNN	PROPOSED1	PROPOSED 2 & WNN
RPE	13.12	12.03	3.84	3.61	3.57
MAPE	8.84	6.16	5.12	5.01	5.01
RMSE	7.24	5.82	5.72	5.20	5.11
RUNNINGTIME	12.48	1.74	15.72	13.01	13.96

5. Заключение

В проведенном исследовании первоначально в качестве индекса оценки был выбран почасовой показатель скорости. Основываясь на выполненном исследовании, было выявлено, в какое время в городе Хамедан возрастает количество пробок, а движение транспортных средств замедляется. Было определено, что в среднем в течение дня три раза наблюдается замедление движения транспортных средств и увеличение плотности трафика. В часы пик утром, днем и вечером на дорогах возникают задержки и заторы, длящиеся около двух часов.

На основе анализа географии сети основных дорог Хамедана была повышена точность получаемых данных и улучшено понимание ситуации с сетевым трафиком в Хамедане, что создает основу для управления трафиком в будущем.

Одновременно на основе использования нейронной сети была выявлена связь между снижением скорости городских транспортных средств и увеличением плотности движения. Полученные результаты показали, что обучение сети на упорядоченных данных дает лучшие результаты при прогнозировании среднесуточной загруженности, нежели такое обучение на случайных данных. При этом в случае комбинирования генетического алгоритма и нейронной сети скорость анализа и точность прогнозов возрастают, а ошибки снижаются.

Выявление мест на городских дорогах, где возникают заторы при движении транспорта, также помогает идентифицировать пробки города и регистрировать связанные с ними события в электронных системах города, проводить изменения топологии дорог и строить новые дороги там, где это необходимо для улучшения условий движения. Авторам удалось выявить условия, возникающие на дорогах города Хамедан в часы пик. В дальнейшем можно будет устранить проблемы движения в городе, можно будет так вести планирование трафика, чтобы минимизировать длительность заторов и улучшить эту ситуацию в городе.

Список литературы / References

[1] Chourabi H., Nam T., Walker S., Gil-Garcia J.R., Mellouli S., Nahon K., Pardo T.A., Scholl J. Understanding smart cities: An integrative framework. In Proc. of the 45th Hawaii International Conference on System Science, 2012, pp.2289-2297.

[2] Rashid B., Rehmani M.H. Applications of wireless sensor networks for urban areas: A survey. Journal of Network and Computer Applications, vol. 60, 2016, pp. 192 –219.

[3] Sahoo J., Cherkaoui S., Hafid A. A novel vehicular sensing framework for smart cities. In Proc. of the 39th Annual IEEE Conference on Local Computer Networks, 2014, pp. 490–493.
[4] Al-Turjman F., Hassanein H., Ibnkahla M. Efficient deployment of wireless sensor networks targeting environment monitoring applications. Computer Communications, vol. 36, issue 2, 2013, pp. 135–148.
[5] Al-Turjman F. Cognitive-node architecture and a deployment strategy for the future sensor networks. Mobile Networks and Applications, vol. 24, issue 5, pp. 1663-1681.
[6] Geisler S., Quix C., Schiffer S., Jarke M. An evaluation framework for traffic information systems based on data streams. Transportation Research, Part C: Emerging Technologies, vol. 23, 2012, pp. 29-55
[7] Anthopoulos L., Fitsilis P. From Online to Ubiquitous Cities: The Technical Transformation of Virtual Communities. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 26, 2009, pp. 360-372.
[8] Shwayri S.T. A model Korean ubiquitous eco-city? The politics of making Songdo. Journal of Urban Technology, vol. 20, issue 1, 2013, pp. 39-55.
[9] Sanchez L. et al. SmartSantander: IoT experimentation over a smart city testbed. Computer Networks, vol. 61, 2014, pp. 217-238.
[10] Bakici T., Almirall E., Wareham J. A smart city initiative: the case of Barcelona. Journal of the Knowledge, vol. 4, issue 2, 2013, pp. 135-148.
[11] Jia S., Peng H., Liu S. Urban traffic state estimation considering resident travel characteristics and road network capacity. Journal of Transportation Systems Engineering and Information Technology, vol. 11, issue 5, pp. 81-85.
[12] Zhu F. Research on index system of urban traffic congestion measures. Master Degree thesis. Nanjing, Jiangsu, China, Southeast University. 2006, pp. 4-15 (in Chinese).
[13] Quiroga C.A. Performance measures and data requirements for congestion management systems. Transportation Research, Part C: Emerging Technologies, vol. 8, issue 1, 2000, pp. 287-306.
[14] Robert R., Theodore F. Contrasting the Use of Time-Based and Distance-Based Measures to Quantify Traffic Congestion Levels: An Analysis of New Jersey Counties. In Proc. of the 81th Annual Meetings of the Transportation Research Board, 2002.
[15] Bertini R.L., Leal M., Lovell D.J. Generating Performance Measures from Portland's Archived Advanced Traffic Management System Data. In Proc. of the 81th Annual Meetings of the Transportation Research Board, 2002.
[16] Bertini R.L., Tantiyanugulchai S. Transit buses as traffic probes: Use of geolocation data for empirical evaluation. Journal of the Transportation Research Board, vol. 1870, issue 1, 2004, pp. 35-45.
[17] Coifman B., Kim S.B. Measuring freeway traffic conditions with transit vehicles. Journal of the Transportation Research Board, vol. 2121, issue 1, 2009, pp. 90-101.
[18] Houli D. et al. Network-wide traffic state observation and analysis method using pseudo-color map. Journal of Transportation Systems Engineering and Information Technology, vol. 9, issue 4, 2009, pp. 46-52.
[19] Turochy R.E., Smith B.L. Measuring variability in traffic conditions by using archived traffic data. Journal of the Transportation Research Board, vol. 1804, issue 1, 2002, pp. 168-172.
[20] Wang Y., Papageorgiou M., Messmer A. Real-time freeway traffic state estimation based on extended Kalman filter: Adaptive capabilities and real data testing. Transportation Research, Part A: Policy and Practicem, vol. 42, issue 10, 2008, pp. 1340-1358.
[21] Vlahogianni E.I., Karlaftis M.G., Golias J.C. Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. Transportation Research, Part C: Emerging Technologies, vol. 13, issue 3, 2005, pp. 211-234.
[22] Yin H., Wong S., Xu J., Wong C.K. Urban traffic flow prediction using a fuzzy-neural approach. Transportation Research, Part C: Emerging Technologies, vol. 10, issue 2, 2002, pp. 85-98.
[23] Lv Y., Duan Y., Kang W., Li Z., Wang F.Y. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, vol. 16, issue 2, 2015, pp. 865-873.
[24] Smith B.L., Demetsky M.J. Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches. In Proc. of the IEEE International Conference on Systems, Man and Cybernetics, vol. 2, 1994, pp. 1706-1709.
[25] Chen D. Research on traffic flow prediction in the big data environment based on the improved RBF neural network. IEEE Transactions on Industrial Informatics, vol. 13, issue 4, 2017, pp. 2000-2008.

[26] Ma X., Tao Z., Wang Y., Yu H., Wang Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research, Part C: Emerging Technologies, vol. 54, 2015, pp. 187-197.
[27] Abdulhai B., Porwa H., Recker W. Short-term traffic flow prediction using neuro-genetic algorithms. Journal-Intelligent Transportation Systems Journal, vol. 7, issue 1, 2002, pp. 3-41.
[28] Massobrio R., Nesmachnow S., Tchernykh A. et al. Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Programming and Computer Software, vol. 44, issue 3, 2018, pp. 181–189.
[29] Alarcon-Aquino V., Barria J.A. Multiresolution FIR neural-network-based learning algorithm applied to network traffic prediction. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 36, issue 2, 2006, pp. 208-220.
[30] Chen Y., Yang B., Meng Q. Small-time scale network traffic prediction based on flexible neural tree. Applied Soft Computing, vol. 12, issue 1, 2012, pp. 274-279.
[31] Fabbiani E., Nesmachnow S., Toutouh J. et al. Analysis of Mobility Patterns for Public Transportation and Bus Stops Relocation. Programming and Computer Software, vol. 44, issue 6, 2018, pp. 508-525.
[32] Yang H. J., Hu X. Wavelet neural network with improved genetic algorithm for traffic flow time series prediction. Optik, vol. 127, issue 19, 2016, pp. 8103-8110.
[33] Lu B., Huang M. Traffic flow prediction based on wavelet analysis, genetic algorithm and artificial neural network. In Proc. of the 2009 International Conference on Information Engineering and Computer Science, 2009, pp. 1-4.
[34] Yi J., Prybutok V.R. A neural network model forecasting for prediction of daily maximum ozone concentration in an industrialized urban area. Environmental Pollution, vol. 92, issue 3, 1996, pp. 349-357.
[35] Grander M.W., Dorling S.R. Neural network modeling and prediction of hourly NOx and NO2 concentrations in urban air in London. Atmospheric Environment, vol, 33, issue 5, 1999, pp. 709-719.
[36] Mobley B.A. et al. Predictions of coronary artery stenosis by artificial neural network. Artificial Intelligence in Medicine, vol. 18, issue 3, 2000, pp. 187-203.
[37] Boone J.M. X-ray spectral reconstruction from attenuation data using neural networks. Medical Physics, vol. 17, issue 4, 1990, pp. 647-654.
[38] IBM Traffic Prediction Tool – IBM Research. Available at: https://researcher.watson.ibm.com/researcher/view_group_subpage.php?id=1248
[39] Как работают Яндекс.Пробки / How Yandex.Traffic Works, available at: https://yandex.ru/company/technologies/yaprobki/ (in Russian).
[40] Tang T.Q., Yi Z.Y., Zhang J., Wang T., Leng J.Q. A speed guidance strategy for multiple signalized intersections based on car-following model. Physica A: Statistical Mechanics and its Applications, vol. 496, 2018, pp. 399–409.
[41] Moretti F., Pizzuti S., Panzieri S., Annunziato M. Urban traffic flow predicting through statistical and neural network bagging ensemble hybrid modelling. NeuroComputing, vol. 167, 2015, pp. 3–7.
[42] Hou Q., Leng J., Maa G., Liu W., Cheng Y. An adaptive hybrid model for short-term urban traffic flow prediction. Physica A: Statistical Mechanics and its Applications, vol. 527, 2019, article 121065.

Информация об авторах / Information about authors

Мехди ШИРМОХАММАДИ получил степень магистра в области информационных технологий в Казвиноском филиале Исламского университета Азад, он является аспирантом в области компьютерных программных систем в Аракском филиале Исламского университета Азада. В настоящее время он преподает на факультете компьютерной инженерии Хамеданского филиала Исламского университета Азад. Его основные научные интересы: сенсорные сети и системы принятия решений.

Mehdi SHIRMOHAMMADI received his MS degree in Information Technology from Islamic Azad University, Qazvin Branch, Qazvin, Iran and he is PhD student in Computer Engineering-Software systems in Islamic Azad University, Arak Branch, Arak, Iran. He is currently lecturer at Department of Computer Engineering, Islamic Azad University, Hamedan Branch, Hamedan, Iran. His main research interest is Sensor networks and decision systems.

Мансур ЭСМАИЛПУР получил степень кандидата компьютерных наук в Национальном университете Малайзии, Малайзия, в 2011 году. Он является доцентом кафедры вычислительной техники и разработки программного обеспечения отделения Хамеданского

филиала Исламского университета Азад. Он активно консультирует промышленность и участвует в нескольких проектах по исследованию и передаче технологий, осуществляемых в сотрудничестве с промышленными партнерами. Его исследовательские интересы включают архитектуру программного обеспечения, электронное обучение, интеллектуальный анализ данных и системы обучения. Он является членом IEEE, IEEE Computer Society и ACM.

Mansour ESMAEILPOUR received the PhD degree in Computer Science from the National University of Malaysia, Malaysia, in 2011. He is an Assistant professor of Computer Engineering, Software Engineering Department of the Islamic Azad University Hamedan Branch, Hamedan, Iran. He is actively consulting in industry and has been involved in several research and technology transfer projects conducted in cooperation with industrial partners. His research interests include software architecture, e-learning, data mining and learning systems. He is a member of the IEEE, the IEEE Computer Society, and ACM.

DOI: 10.15514/ISPRAS-2020-32(3)-13

Использование компьютерных методов и систем в изучении права, интеллектуальном анализе и моделировании правовой деятельности: систематический обзор

[1] *Е.В. Трофимов, ORCID: 0000-0003-4585-8820 <diterihs@mail.ru>*
[2] *О.Г. Мецкер, ORCID: 0000-0003-3427-7932 <olegmetsker@gmail.com>*
[1] *Всероссийский государственный университет юстиции (РПА Минюста России), 117638, Россия, г. Москва, ул. Азовская, д. 2, корп. 1*
[2] *Национальный медицинский исследовательский центр имени В. А. Алмазова, 197341, Россия, г. Санкт-Петербург, ул. Аккуратова, д. 2*

Аннотация. Интеграция вычислительных систем и методов в юридическую деятельность позволяет извлечь такие выгоды, как ресурсосбережение, повышение объективности, полноты и точности интеллектуальных результатов. Понимание основных научных достижений и тенденций на стыке компьютерных и правовых наук акцентирует внимание на перспективных научно-технологических направлениях информатизации права. Настоящий обзор систематизирует важнейшие достижения на стыке правовых и компьютерных наук, охватывая зарубежные и отечественные научные публикации за 1949–2020 годы. Исследования компьютерных методов и систем, инициированные американской юриметрикой, были направлены на хранение, индексацию, абстрагирование и поиск юридических текстов и привели к созданию правовых информационно-поисковых систем, тогда как отечественная правовая кибернетика стала пионером в сфере автоматизации криминалистической экспертизы. На базе методов искусственного интеллекта развилось компьютерное моделирование юридических рассуждений, сместившееся в дальнейшем в область юридического диалога и конфликта правовых аргументов, а в последние годы трансформирующего полученный опыт на основе современных компьютерных моделей, шаблонов и архитектур. Популяризация систем для поддержки принятия решений обеспечила мультизадачность систем, включающую информационный поиск, юридическую аргументацию, аналитику, прогноз и контроль. Новейшие сегменты исследований ориентированы на применение методов машинного обучения и интеллектуальный анализ больших данных. Практически все успешные методологические решения сохраняют свое значение, продолжая применяться непосредственно или послужив основой для дальнейшего развития вычислительных методов и информационных систем в правовой деятельности.

Ключевые слова: право и искусственный интеллект; юриметрика; правовая информатика; кибернетические юридические модели; правовые экспертные системы; интеллектуальное управление документами; большие данные; машинное обучение; правовые системы, основанные на знаниях; прикладные юридические онтологии

Для цитирования: Трофимов Е.В., Мецкер О.Г. Использование компьютерных методов и систем в изучении права, интеллектуальном анализе и моделировании правовой деятельности: систематический обзор. Труды ИСП РАН, том 32, вып. 3, 2020 г., стр. 147-170. DOI: 10.15514/ISPRAS–2020–32(3)–13

Благодарности. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-111-50534.

Application of Computer Techniques and Systems in the Study of Law, Intellectual Analysis and Modeling of Legal Activity: A Systematic Review

[1] *E.V. Trofimov, ORCID: 0000-0003-4585-8820 <diterihs@mail.ru>*
[2] *O.G. Metsker, ORCID: 0000-0003-3427-7932 <olegmetsker@gmail.com>*
[1] *All-Russian State University of Justice,*
2 Bldg. 1, Azovskaya Str., Moscow, 117638, Russia
[2] *Almazov National Medical Research Centre,*
2, Akkuratova Str., St. Petersburg, 197341, Russia

Abstract. Integration of computing systems and methods in legal activity allows to extract benefits such as resource saving, increase objectivity, completeness and accuracy of intellectual results. Understanding of the main scientific achievements and trends at the intersection of computer and legal sciences focuses on promising scientific and technological areas of informatization of law. This review systematizes the most important achievements at the intersection of legal and computer sciences, covering foreign and domestic scientific publications for 1949–2020. Researches of computer methods and systems initiated by American jurimetrics were aimed at storing, indexing, abstracting and searching for legal texts and led to the creation of legal information retrieval systems. Domestic legal cybernetics became a pioneer in the field of automation of criminalistics expert examination. Computer modeling of legal reasoning developed on the techniques of artificial intelligence, later it shifted to the field of legal dialogue and conflict of legal arguments, but in recent years it transforms the acquired experience on the basis of modern computer models, patterns and architectures. The popularization of decision-making support systems has provided multi-tasking systems, including information retrieval, legal reasoning, analytics, predictions and control. The latest research segments are focused on the application of machine learning and big data processing. Almost all successful methodological solutions retain their significance, continuing to be applied directly or as the basis for the further development of computational methods and information systems in legal activity.

Keywords: law and artificial intelligence; jurimetrics; legal informatics; cybernetic legal models; legal expert systems; intelligent document management; big data; machine learning; legal knowledge-based systems; legal ontologies

For citation: Trofimov E.V., Metsker O.G. Application of Computer Techniques and Systems in the Study of Law, Intellectual Analysis and Modeling of Legal Activity: A Systematic Review. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 3, 2020. pp. 147-170 (in Russian). DOI: 10.15514/ISPRAS–2020–32(3)–13

Acknowledgments. The reported study was funded by RFBR, project number 19-111-50534.

1. Введение

За последние несколько десятилетий информационно-коммуникационные технологии распространились на различные сферы общественной жизни, включая деятельность юристов. После создания компьютеров были инициированы исследования по внедрению вычислительных средств, технологий, систем и методов в юридическую сферу с тем, чтобы извлечь из автоматизации такие выгоды, как ресурсосбережение, повышение объективности, полноты и точности интеллектуальных результатов.

В настоящей статье приведен систематический обзор важнейших направлений научного поиска, ознаменовавших прогресс в разработке компьютерных методов и систем для изучения права, интеллектуального анализа данных и моделирования правовой деятельности. Впервые подобный обзор охватывает как зарубежные, так и отечественные научные публикации. В статье раскрываются методологические достижения XX века, на которых базируются современные исследования и системы, а также демонстрируется разнообразие методологических подходов к разработкам и экспериментам XXI века, значимость которых еще предстоит оценить.

Понимание достижений и тенденций на стыке компьютерных и правовых наук позволяет акцентировать внимание на наиболее перспективных научно-технологических областях и направлениях. Настоящий обзор, как надеются авторы, поможет развивать идеи информатизации права, принимая обоснованные стратегические решения при планировании исследований, проектировании информационных систем, разработке и реализации правовой политики.

2. Юриметрика и правовой информационный поиск

Идейным основателем юриметрики как научного направления и создателем самого термина «юриметрика» стал Ли Ловенгер, который написал одноименную статью программного характера в 1949 г. и заявил в ней, что использование современной научной методологии и технологии (в том числе кибернетической) должно стать следующим шагом в развитии права. Компьютерные методы и системы, учитывая их тогдашний уровень, воспринимались юриметрикой больше как перспективные (в обозримом будущем) инструментальные средства для повышения производительности вычислений. Ловингер, в частности, ставил вопрос о создании машины для разрешения судебных дел, наивно полагая, что сложность этой задачи состоит в отсутствии юридических терминов, которые, подобно цифрам и символам, можно ввести в машину [71].

С конца 1950-х г. идеи юриметрики и внедрения кибернетики в право стали популярны не только в Америке, но и в Европе. Так, Люсьен Мель, продвигая идеи интеграции логики, Булевой алгебры и двоичной записи, предложил создание четырех типов «юридических машин»: 1) «информационной машины» для поиска правовой информации, предоставляющей релевантные элементы информации, создание которой предполагалось путем систематизации (кодификации) правовой информации, определения базовых юридических понятий по типу ключевых слов и колонтитулов и представления данных, понятий, ситуаций и проблем в бинарной форме; 2) «консультационной машины», выполняющей на основе логики и Булевой алгебры концептуальный и реляционный анализ и дающей точный ответ на поставленный перед ней юридический вопрос; 3) машины для проверки логической согласованности правовых положений законов или конвенций; 4) машины для перевода юридических текстов [80].

Уже в начале своего пути юриметрика развивала идеи компьютерного хранения и поиска информации, поскольку это был насущный экономический вопрос, связанный с сокращением операционных затрат на работу с запредельными для ручного поиска объемами информации. В середине XX в. основным методом организации поиска юридических документов была «ручная» индексация документов с последующим «ручным» просмотром индексов. Развитие компьютерных систем и методов поставило на обсуждение три основных подхода к компьютерному решению этой проблемы: 1) перевод индексов и поиска по ним в электронную форму; 2) компьютерное совершенствование самого метода индексации; 3) переход от индексации к полнотекстовому электронному поиску. Все три направления, хотя и по-разному, дали результаты, были восприняты в дальнейших исследованиях и в определенной мере используются до сих пор.

2.1. Эксперименты Моргана

Самым очевидным стал перевод в электронный вид бумажных индексов и поиска по ним. Подход обеспечивал индексацию документов путем их экспертного абстрагирования, а поисковый запрос составлялся на основе экспертно сформированного индекса. В конце 1950-х гг. в Университете штата Оклахома Роберт Морган использовал компьютер для индексного поиска, назвав его подходом «правового вопроса».

Реферированная юридическая библиотека переводилась в электронный формат, чтобы обеспечить поиск по рефератам. Эксперт осуществлял спецификацию путем анализа фактов

конкретного дела для определения сути правового вопроса, формулировал его в виде юридически осмысленного концепта (слова, фразы, параграфа и т.п.) и индексировал запрос. По цифровому индексу компьютер выдавал список релевантных юридических источников, требовавших уже экспертного отбора [86].

2.2. Эксперименты Мелтон и Бенсинга

В 1959–1960 гг. Университете Западного резервного района (г. Кливленд) в эксперименте Джессики Мелтон и Роберта Бенсинга к юридическим текстам был применен метод «семантически кодированного реферирования», стратегической целью которого являлось создание новых знаний на основе известных. Суть подхода сводилась к абстрагированию текстов, индексации, нормализации языка (терминологии и синтаксиса) и использованию машинного тезауруса для хранения и поиска текстов. Разработчиками были закодированы статья 2 Единообразного торгового кодекса США 1952 г. и несколько судебных решений.

Основу эксперимента составлял специально созданный семантический код, представлявший собой сочетание тезауруса и многоразмерной классификации научной терминологии, в основе которой находились индексные термины, отобранные и классифицированные экспертным путем. Значения терминов (слов, выражений) кодировались «семантическими факторами» и «численными факторами», а отношения между терминами – «ролевыми индикаторами» и знаками пунктуации.

Кодировка имела сходство с универсальной десятичной классификацией П. Отле и А. Лафонтена, но при этом нормализация языка и использование машинного тезауруса давали возможность осуществлять поиск не только по концептам (абстрактным значениям), но и по конкретным терминам, а также в отчасти учитывать общий смысл поискового запроса и абстрагированного документа. Процедурно подход предусматривал кодирование фрагментов текста и поисковых запросов с последующей идентификацией реферата, содержащего коды или элементы кодов, которые соответствуют логической конфигурации поискового запроса [81, 82].

2.3. Эксперименты Хорти и Кела

В 1958–1962 гг. в Университете Питтсбурга под руководством Джона Хорти и Уильяма Кела проводилось исследование по электронному хранению, упорядочению юридических материалов и поиску релевантной информации, получившее названия подхода «ключевых слов в комбинации» или «питтсбургской системы». В ходе исследования статуты штата Пенсильвания объемом 6230529 слов (включая 2815340 общих слов), объединенных в 31113 секций статутов (документов), были набраны, проверены, исправлены вручную, переведены в электронный вид. Из встречавшихся в статутах 24 тыс. слов (приведенных к начальным формам), исключая имена собственные и 112 общих слов, не использовавшихся в поиске, был составлен алфавитный словарь, а уже из него был создан юридический тезаурус, на основе которого составлялся поисковый запрос для более точной постановки поисковой задачи.

Поиск обеспечивался по отдельным словам или их комбинациям. Казуальная проверка точности поиска показала, что «ручной» поиск выявил 42,6% релевантных статутов против 97,9% в компьютерном поиске, однако последний выдавал также интенсионально нерелевантные ответы и потому требовал экспертного анализа [45, 46, 49]. Пионерские работы Хорти приобрели всемирную известность и послужили теоретическим базисом для дальнейших изысканий в области полнотекстового поиска.

2.4. Информационно-поисковые системы индексного типа

В дальнейшем основу электронных индексов были положены разработанные в общем виде Хансом Луном индексация KWIK (ключевое слово в контексте) и SDI (выборочное

распространение информации), которые во многом были восприняты и в полнотекстовом поиске. Так, в 1964–1966 гг. в проекте «Law Research Services» под руководством Эллиаса Хоппенфельда был реализован индексный подход на картотеке в объеме свыше 1 млн. абстрагированных судебных решений, а поиск обеспечивался на основе дискрипторов и Булевых операторов [44].

В 1967 г. в Бельгии был создан малобюджетный правовой индекс CREDOC, основанный на SDI и методе ретроспективного поиска; базу CREDOC составили 60 тыс. документов, а тезаурус включал 6500 первичных концептов, которые в сочетании с 50 базовыми ключевыми словами и 500 ключевыми дескрипторами давали 31 тыс. терминов [99].

2.5. Полнотекстовые информационно-поисковые системы

В середине 1960-х гг. несколько исследовательских групп в США, Канаде и Великобритании приступили к разработке поисковых систем по полнотекстовым правовым документам.

Наиболее известный опыт, благодаря быстрому расширению информационной базы и коммерциализации результата, – это поисковая система OBAR, исследования по созданию которой проводились в 1964–1967 гг., а в 1968–1970 гг. перешли на этап разработки реально работающей системы интуитивно понятного поиска по ключевым словам, оперировавшего на массиве полнотекстовых законодательных и судебных документов штата Огайо [43, 128]. К 1973 г. расширенная система, переименованная в LEXIS, охватывала право нескольких штатов и ряд отраслей федерального права США в объеме около 600 млн. символов [106]. Однако LEXIS еще не имела автоматического тезауруса, предлагающего синонимы, антонимы, обобщения, грамматические или орфографические варианты выбранных поисковых терминов, а логика поиска была ограничена Булевыми операторами, дополненными указателями расстояния и направления конъюнкции [120].

В Великобритании компьютерные исследования в области правовых поисковых систем, вдохновленные работами Хорти и известные как «оксфордские эксперименты», велись с 1961 г. юристами и статистиками под руководством Колина Таппера [121]. Благодаря их успехам в 1968–1969 гг. Брайан Ниблетт и Норман Прайс разработали на языке Fortran систему STATUS, обеспечивавшую компьютерный поиск по статутному праву Соединенного Королевства в области атомной энергетики (около 150 тыс. слов) на основе частотного словаря и индексирования типа KWIC [90].

В 1967–1973 гг. в Университете Куинс в Кингстоне под руководством Хью Лоуфорда разрабатывался поисковый сервис QUIC/LAW, который включил базу данных из полнотекстовых пересмотренных статутов Канады на английском и французском языках, неофициальную консолидацию федеральных приказов и правил, базу данных из полнотекстовых судебных решений и отчетов и две научные базы данных, содержавшие свыше 67 тыс. избранных рефератов с библиографическими записями [65]. В 1968 г. коллектив под руководством Жака Буше и Эджана Макайя начал работу над автоматическим правовым поисковым сервисом по прецедентам DATUM, и к 1971 г. был создан банк полнотекстовых судебных актов объемом около 140 млн. символов; методологическую основу системы составили двуязычный тезаурус и Булева логика, а поиск осуществлялся на английском и французском языках, обеспечивая благодаря тезаурусу учет синонимичных и более общих терминов [14, 73].

Подобные разработки велись и в странах континентальной Европы: например, в Швеции к 1972 г. уже была создана правовая информационно-поисковая система IMDOC, затем внедренная также в Финляндии под названием MINTTU [66], а в ФРГ с начала 1970-х г. активно разрабатывалась информационно-поисковая система JURIS [16].

2.6. Методы усовершенствования информационно-поисковых систем

Ряд достижений в вышеуказанные системы не были интегрированы, несмотря не проводившиеся исследования. Так, в конце 1960-х гг. Колин Таппер ставил эксперименты поиска по массиву судебных актов с использованием внутритекстовых ссылок на прецеденты [119], а с 1965 г. проектная группа Уильяма Элдриджа работала над улучшением автоматизированного индексирования и поиска за счет математического моделирования частотности поисковых терминов и анализа их статистической значимости [28].

Поэтому после внедрения успешных информационно-поисковых систем велись исследования по улучшению поиска на основе ключевых слов, статистического и семантического подходов, а также разнообразных методов, среди которых Булева логика, регулярные выражения, примерное совпадение строк, кластеризация, частотный анализ, тезаурус, парсинг и понимание естественного языка, формирование гипотез, концептуальное представление, сопоставление по правилам, различные виды рассуждений и т.д.

Пожалуй, наиболее методологически значимыми направлениями в этой области, выходящими за далеко рамки поисковых задач, стали концептуальное моделирование и модели представления знаний, которые в некоторых исследованиях [130] даже стали объединяться.

2.7. Концептуальное моделирование

Использование концептуального моделирования направлено на достижение языковой выразительности и позволяет включать в систему концептов единичные факты, извлекаемые из вводимой информации, и тем самым, в частности, решать проблему синонимичности [77], но основная проблема такого подхода состоит в обеспечении самообновления для системы концептуально организованного поиска, позволяющего интегрировать новые концепты [110], что в области права чрезвычайно актуально.

В конце 1970-х гг. Кароль Хафнер на концептуальной основе статей 3 и 4 Единообразного торгового кодекса США построила систему LIRS с базой знаний в виде семантической сети, содержащей около 300 вершин и позволявшей вести поиск по информационному массиву из 186 судебных дел, 110 секций кодекса и 188 официальных разъяснений [42].

В 1980-х гг. этот подход был усложнен другими методами и представлен, например, в американской системе RUBRIC, включавшей сложные правила рассуждений на основе критериев, концептов и неопределенных интервальных значений с использованием технологий искусственного интеллекта [123-125], а также в норвежской системе ARCTIS с тезаурусом, отражающим структуру нормативного массива [12].

2.8. Модели представления юридических знаний и современные проблемы

Модели представления юридических знаний обеспечивают интеллектуализацию информационно-правового поиска. Так, некоторые ученые выстраивают поисковые модели, ориентируясь на классификацию и факторизацию [138]. Другие ученые разрабатывают предметно-ориентированные онтологии, обеспечивающие более высокий уровень выразительности, чем тезаурус [40, 108]. Предпринимались попытки выстраивать юридические онтологии на основе языка описания онтологий для семантической паутины (OWL): например, в 2005 г. в предметно-ориентированной онтологии для права интеллектуальной собственности RDDOnto [30].

Но и сейчас, как и в ранних экспериментах, обработка и ввод исходных данных в правовые информационно-поисковые системы требуют значительных затрат экспертного труда. Кроме того, став сервис-ориентированными, более полными и точными за счет использования комплексов логических, математических, статистических и компьютерных методов и

облегчив рутинную работу юристов, эти системы пока не обеспечивают пользовательской простоты и высокой степени полноты и точности поиска для интенсионально сложных запросов.

3. Отечественная правовая кибернетика

В СССР в 1957 г. Л. Г. Эджубов начал работу над автоматизацией системы дактилоскопической регистрации, а Д. А. Керимов инициировал работу над созданием специального информационного языка в области права [54]. Эти два направления и задали векторы развития компьютерных систем и методов в правовой сфере, поскольку в решении большинства задач «юридической кибернетики» (названной так академиком А. И. Бергом в 1962 г.) не было методологического прогресса. Например, предпринимались попытки выработать подход к алгоритмизации (программированию) юридического процесса и доказывания [20, 62, 129], но их вычислительная составляющая была крайне слаба, хотя и по сей день [57] делаются попытки реанимации этой идеи.

3.1. Криминалистические автоматизированные информационные системы

Реализованный Л. Г. Эджубовым совместно с С. А. Литинским компьютерный метод решал проблему работы с большим объемом регистрационного материала за счет кодового представления отпечатков и пальцевых следов. В основу нового подхода, вместо формулярного, были положены учет однородного частного признака (местоположения деталей папиллярного узора на поле отпечатка), координатное кодирование и идентификация на основе зонально-точечного сравнения отпечатков и пальцевых следов. В рамках нового подхода два авторских коллектива уже в 1957–1959 гг. подали шесть заявок на авторские изобретения, в 1959 г. состоялись испытания экспериментального образца фотоэлектронной модели дактилоскопического автомата, а в 1960 г. был создан промышленный образец специализированной дактилоскопической ЭВМ «Минск-100» [26, 100]. На тот момент в области автоматизации дактилоскопической экспертизы СССР более чем на десятилетие опередил другие страны, где аналогичный подход к разработке АДИС (APIS) только к 1969 г. был проработан Джозефом Уэгстейном [136].

Основной компьютерной проблемой в правоохранительном сегменте считалась автоматизация процедуры идентификации лиц и объектов, и она решалась разработкой специальных алгоритмов и использованием статистических и вероятностных методов [32, 95, 112], а основными задачами стали точность, скорость и полнота исследования. Например, в 1964 г. Литовский НИИСЭ совместно с лабораторией теоретической кибернетики ЛГУ провел успешные эксперименты в области почерковедческой экспертизы, положив в основу решения задач идентификации и дифференциации почерковых объектов аппроксимационный подход по принципу «обучающейся машины», которая при распознавании графических образов не использует заранее заданные признаки, а «вырабатывает» их сама, обучаясь распознаванию на тренировочной последовательности по рекуррентному конечно-сходящемуся алгоритму решения систем неравенств [60, 61].

После 1970 г., когда автоматизация постепенно проникла в правоохранительную систему, исследования вновь активизировались и распространились на автороведческую [134], портретную [140], судебно-автотехническую [27], трасологическую [39] и другие виды экспертиз (исследований), а также на различные виды учетов: криминалистических [92], административных [113] и оперативных [36]. Однако, не считая внедрения очевидных для своего времени технических решений (не специфичных для предметной области права), только к концу 1980-х гг. ведомственной наукой были получены значимые методологические результаты в области компьютерных систем [56].

3.2. Юридический информационный язык и информационный поиск

Важным направлением стала автоматизация юридической деятельности с использованием специально созданного информационного языка, адаптированного для компьютерных вычислений. Эти работы первоначально велись на юридическом факультете ЛГУ, в том числе совместно с экспериментальной лабораторией машинного перевода [4], но, в отличие от юриметрики, решавшей поисковую задачу для правоприменения, отечественные ученые ставили цель совершенствования правотворчества [50].

В первые несколько лет исследований доминировали идеи, похожие на подход Мелтон и Белсинга, но сочетавшиеся с тезисом Меля о полной систематизации правовой информации: для начала предлагалось создать специальный формализованный язык, стандартизировать и классифицировать юридические концепты, отредактировать и формализовать юридический материал на основе общих принципов символизации, словаря терминов, логических и грамматических связок, а потом разработать алгоритмы решения разнообразных юридических задач [51]. Исследования в области информационного поиска, включая разработку информационного языка [48], в начале 1960-х гг. считались успешными и, как и эксперименты Хорти, демонстрировали намного большую точность автоматического правового поиска, по сравнению с «ручным»: 3% пропусков против 53% соответственно [55]. Однако попытка использовать язык-посредник между юридическим и машинным языком уже тогда диссонировала с аргументами Йегошуа Бар-Хиллела, которые дискредитировали в мировой науке идею корректного машинного перевода.

В 1964 г. было заявлено, что кибернетика не пригодна для систематизации права, в отличие от решения информационно-поисковых задач [52], и исследователи переключились на проблему автоматизации справочно-информационной юридической службы [53]. Оба намеченных пути: простое абстрагирование нормативных актов с выявлением в их содержании ограниченного количества основных терминов для решения поисковой задачи (подобно опыту Моргана) и методологически проблемная разработка правового информационно-логического языка с универсальными возможностями для логических операций над юридическим материалом [114], – оказались провальными, и к началу 1970-х гг. исследовательская задача была редуцирована [137] до западного опыта, где уже были созданы полнотекстовые поисковые системы.

3.3. Отечественные информационно-поисковые системы

В 1973 г. под руководством С. С. Москвина была разработана специализированная (в области лесного хозяйства) ИПС «Право-1» [10, 29]. С 1976 г. под руководством В. А. Копылова сначала был разработан правовой тезаурус из 5 тыс. дескрипторов с родовидовыми отношениями, а затем в 1982 г. внедрена универсальная АИПС «Законодательство», представлявшая собой модифицированный тип уже известной документальной информационно-поисковой системы. Перед вводом в систему документы проходили предварительную экспертную обработку по принципу свободного индексирования, а созданный экспертами тезаурус автоматически пополнялся при вводе документов, что привело к его увеличению к 1989 г. до 125 тыс. дескрипторов, из которых только для 50 тыс. эксперты задали отношения.

Лингвистическое обеспечение АИПС включало информационно-поисковый язык дескрипторного типа, а поисковое предписание могло содержать ключевые слова, текстовые роли, фактографические данные, связанные логическими операциями (И, ИЛИ, НЕ) и операторами сравнения текстовых полей [133]. В экспериментах по оценке эффективности поиска были получены 81% полноты и 72% точности системного поиска [5], что диссонировало с критериями социально-правовой эффективности АИПС, обоснованными в самом же ВНИИСЗ еще в период разработки этой АИПС [19]. Методологическим венцом этого этапа можно считать докторскую диссертацию В. А. Копылова [58].

В конце 1980-х гг. активизировались технологические разработки и были созданы другие АИПС: «ЮСИС» (1989), «Гарант» (1990), «Закон» и «Кодекс» (1991), «КонсультантПлюс» (1992), «Система» (1993) и т.д. Эти разработки сопровождались немногочисленными (в силу коммерциализации продукта) публикациями, обосновывавшими внедрение в АИПС отдельных достижений точных наук [33, 91, 94].

4. Право и методы искусственного интеллекта

Подходы к внедрению компьютерных методов и информационных систем в предметную область права, получившие в социалистических странах название правовой (юридической) кибернетики, а за рубежом именовавшиеся некоторыми исследователями «юскибернетикой», помимо правовой информатики, которая сконцентрировалась на информационном поиске, включали также направление кибернетических юридических моделей [72]. Успехи компьютерных наук и правовой информатики к 1970-х гг. сделали реальным компьютерное моделирование на основе производительных вычислительных методов, получившее общее название «право и искусственный интеллект». Первенство постановки вопроса в таком сочетании принадлежит Брюсу Бьюкенену и Томасу Хедрику, которые в 1970 г. подняли вопрос о компьютерном моделировании юридических рассуждений [15].

Эти идеи стали основой для разработки правовых экспертных систем, основанных (в самом общем виде) на базах знаний и алгоритмах вывода и предназначенных для накопления и извлечения знаний в узких предметных областях права для экспертного решения различных проблем. Правовые экспертные системы выстраивались по типу дедуктивных рассуждений или прецедентов, усложнялись до гибридных моделей, моделей с нечеткой логикой, интегрировались с нейронными сетями.

4.1. Правовые экспертные системы на правилах или кейсах

В 1972–1973 гг. в США пионер в этой области Л. Торн Маккарти обосновал и разработал первую предметно-ориентированную экспертную систему TAXMAN, позволявшую с использованием формализованных правил подглавы «С» главы I Налогового кодекса США 1954 г. на основе классификации вводимой информации выдавать юридическое обоснование для решения задачи освобождения от подоходного налога в случае некоторых типов реорганизации корпораций [76]. С учетом дескриптивной ограниченности созданной системы и понимания нечеткости и динамичности концептов в судебной практике Маккарти к началу 1980-х гг. разработал на языке AIMDS усовершенствованную систему TAXMAN II, в которой усложнил структуру концептов моделью «прототип + деформация» [78].

С 1984 г. в Массачусетском университете реализовывался проект COUNSELOR по изучению проблем в структуре дискурса и обработке текста в рамках интегрированного интерфейса с сильной экспертной системой. В ходе этого проекта к 1987 г. Эдвиной Риссланд и Кевином Эшли была разработана система HYPO, которая использовала рассуждения по типу прецедентов на основе индексов («измерений») и работала в предметной области нарушения коммерческой тайны. Вместо однозначных ответов HYPO генерировала аргументы для истца и ответчика, сопоставляемые с благоприятными для них случаями [101].

4.2. Гибридные правовые экспертные системы

К 1989 г. Эдвина Риссланд и Дэвид Скалак в гибридной системе CABARET объединили кейс-подход HYPO с модулем по типу правил, обеспечивавшим прямую и обратную цепи рассуждений, а также с эвристическим управлением задачами по созданию аргументов для поддержки конкретной ситуации. Система CABARET была программной оболочкой, выстроенной независимо от домена, а вычислительные эксперименты проводились в области подоходного налогообложения [102].

Сложности в моделировании правовых рассуждений (в частности, в формировании правил, обработке естественного языка, построении баз знаний) приводили к резкому ограничению доменной области, на которой работали разрабатываемые методы и системы. Для преодоления этих недостатков предпринимались попытки ввести в качестве улучшающего элемента нейронные сети. Так, в 1991 г. была предложена архитектура экспертных систем PROLEXS для гетерогенного домена, в котором предлагалось объединить неоднородные источники знаний за счет применения к ним различных методов и языковых представлений. В экспериментах была использована доменная область датского права в области аренды недвижимости, в которой были выделены четыре группы знаний (законодательство, правовые знания, экспертные знания и прецеденты), каждая из которых имела собственное языковое представление и специальный механизм вывода. Система на данной архитектуре была гибридной и использовала рассуждения на основе правил и кейсов, а также нейронные сети для отбора прецедентов [135].

В дальнейшем гибридизация правовых экспертных систем и стремление к их доменной или методологической расширяемости стали общим трендом. Например, в Австралии к 1993 г. Джеймс Поппл обосновал систему SHYSTER, основанную на кейсах, но преобразуемую в гибридную систему при запуске модуля, основанного на правилах. SHYSTER не был связан с доменной областью и должен был демонстрировать принципиальную простоту организации системы, поэтому содержал модули токенизации и парсинга, работавшие на базе прецедентов, и по весу атрибутов вычислял близость рассматриваемого случая известным прецедентам [96]. К 1994 г. в США Кэтрин Сандерс в развитие подходов HYPO и CABARET представила систему с нечеткими правилами CHIRON для решения ограниченных правовых вопросов налогообложения. В этой системе были использованы прототипы (абстрактные консервативные планы) решений в области налогообложения, рассуждения по типам кейсов и правил для модификации (деформации и адаптации) заданных прототипов, а также модальные и интенсиональные логики для выражения фактов и отношений [107].

4.3. Абстрактные правовые аргументы, шаблоны «классной доски» и сервис-ориентированная архитектура

Вместе с тем, в середине 1990-х гг. качестве промежуточного итога исследований было констатировано, что попытки создать алгоритмы, которые могут самостоятельно рассуждать при принятии правовых решений, не увенчались успехом, поскольку моделирование закона и подражание процессам юридического обоснования оказались более сложными и тонкими, чем первоначально предполагалось [2]. Понимание этого факта трансформировало исследования юридических рассуждений и правовой аргументации, и с середины 1990-х гг. правовые экспертные системы все более стали уходить из сферы моделирования правильных (в материально-правовом смысле) суждений.

Благодаря идеям таиландского профессора Фана Данга об абстрактных правовых аргументах, операбельных независимо от методов их генерации [22-24], усилия были перенаправлены в сферу юридического диалога (игр) и выбора (конфликта) правовых аргументов. Этот подход предполагает уровневое представление аргументации и отход от немонотонной логики, а объектно ориентируется на исследование юридического диалога, в том числе в ситуации неправоты всех сторон. Такая парадигма (в юридическом смысле – процессуальная или доказательственная) получила широкое распространение и считается актуальной до сих пор [3, 9, 11, 97, 98].

Кроме того, в настоящее время предпринимаются попытки создания более технологичных систем, построенных на использовании новейших компьютерных достижений в области искусственного интеллекта. Например, получили распространение системы по типу «классной доски», основанные на использовании архитектурной модели итеративного экспертного обновления источников знания. Среди таких систем можно упомянуть

экспериментальную систему FRANK, предназначенную для демонстрации прямых и обратных связей между высокоуровневыми объяснительными целями пользователя и деталями выполнения рассуждений по типу кейсов [103]. А в Великобритании Чарльз Стивенс и его коллеги, создавшие прототип системы JAES, считают перспективным построение систем на основе правовых рассуждений по типу кейсов, соединенных с шаблоном «классной доски» и сервис-ориентированной архитектурой [115].

4.4. Системы поддержки принятия правовых решений и мультизадачность

Ослабление интереса к изучению строгих правовых рассуждений связано также с популяризацией систем для поддержки принятия решений, не продуцирующих императивных самостоятельных рассуждений и выводов. Благодаря этому исследования интеллектуальных правовых систем, которые до 1990-х гг. акцентировались на задачах поиска и рассуждений, стали расширяться в направлении мультизадачности.

В круг исследовательских интересов, помимо информационного поиска и юридической аргументации, стали включаться аналитика, прогноз и контроль. Исследования в этих областях проводились и ранее (например, предиктивная аналитика в судебной области восходит к вычислительным экспериментам К. Таппера конца 1960-х гг.), однако именно в 1990-х гг. быстрая компьютеризация вкупе со стремлением научно обосновать решение повседневных задач создало повышенный интерес к практикоориентированным интеллектуальным правовым системам различного назначения.

4.5. Правовые информационные системы с интеграцией аргументационной функции

В компьютерных системах усилилась интеграция поисковой и аргументационной функций. Так, в середине 1990-х гг. в системе BankXX на основе сети знаний было реализовано сочетание функций эвристического поиска, индексации и правовой аргументации, позволяющее собирать и выстраивать аргументы по базе судебных дел, научных суждений и прототипов фактических сценариев [104, 105], т.е. была построена система поиска и аргументации, помогавшая эксперту в принятии правовых решений.

В 2017 г. была представлена поисковая система LexrideLaw, извлекавшая аргументы из судебных актов или обеспечивавшая к ним доступ через выбор узлов в онтологии судебных споров либо через поиск по ключевым словам в реляционных вопросах [35].

В 2019 г. на архитектуре SaaS (программное обеспечение как услуга) было создано рамочное Web-приложение NAI для нормализации юридических текстов и автоматизированных рассуждений над ними с использованием стандартной деонтической логики [69].

4.6. Правовые информационные системы с интеграцией предсказательной функции

Традиционные задачи поиска и рассуждений стали активно сочетаться с прогнозированием. Например, система SPLIT-UP, которая разрабатывалась в Австралии на протяжении 1990-х гг. Эндрю Страньери и Джоном Зелезникоу, сочетала в себе правовую аргументацию с предиктивной аналитикой. Методологическую основу системы, функционировавшей в доменной области раздела имущества супругов, составили рассуждения по правилам, формализованным из австралийского Закона о семейном праве 1975 г., а также нейронные сети, обученные на стандартных судебных делах. Исследователи ставили перед собой задачу автоматизировать правовые рассуждения для доменной области, в которой значительна правоприменительная дискреция, препятствующая определению моделей судебных рассуждений. В SPLIT-UP за счет интеграции разных методологических подходов

предпринималась попытка учета субъективного фактора в судебном правоприменении (как и, например, в гибридной системе CHIRON). Однако, как и любая система с использованием нейронных сетей, SPLIT-UP испытывала трудности с формированием массива данных для машинного обучения и интеграцией разных методологических парадигм [116, 117].

В 2009 г. Кевин Эшли и Стефани Брюнингхауз представили программу SMILE+IBP, которая для доменной области присвоения коммерческой тайны выдавала прогноз решения заданного случая посредством извлечения фактов уже разрешенных судебных дел и их факторизации на основе классификационных концептов, отражающих стереотипные фактологические паттерны [6].

В 2017 г. в эксперименте по предсказанию исхода дела о предоставлении убежища, основанному на ретроспективном анализе и факторизации обстоятельств ранее разрешенных дел, была достигнута прогнозная точность 80% [25], а в другом эксперименте на массиве дел в области присвоения коммерческой тайны была создана система VJAP, которая для каждого случая формировала граф аргументов и обосновывала прогноз решения конкретного случая, используя схемы аргументов и количественные весовые коэффициенты, извлеченные из предыдущих случаев с применением метода итеративной оптимизации [38].

В 2019 г. для улучшения прогностической функции были проведены эксперименты по экстрагированию и суммаризации судебных текстов с использованием глубокого обучения по сверточным нейронным сетям (CNN) и последующей классификацией. Опыты имели некоторый успех, но показали отсутствие надежного охвата всех аспектов дела и недостаточную адекватность лексических метрик перекрытия для оценки аннотированных текстов [139].

В 2020 г. американскими исследователями была предложена модель, сочетающая стратегии поиска и мультиструктурированное правовое пространство поиска; эта модель рассматривает поиск применимого закона как последовательный процесс принятия решения в определенных поведенческих и когнитивных рамках, что позволяет с некоторым успехом выполнять предсказательную функцию [18].

4.7. Правовые информационные системы с интеграцией функции управления документами, контентом и знаниями

Для решения в общем комплексе также аналитических задач изучаются интеллектуальные инструменты управления правовыми документами, использующими методы обработки естественного языка (NLP), включая нормализацию, векторизацию, кластеризацию и суммаризацию текста. Так, в 1993 г. была представлена система FLEXICON с реализованной в ней моделью структурированного представления знаний в сочетании со статистическим ранжированием, что обеспечивало эффективный менеджмент юридических текстов за счет решения задач поиска и суммаризации текста [34]. В другой системе, KONTERM, были использованы тезаурус, база знаний и экспертные правила, и система автоматически представляла структуру и содержание правового документа, тем самым обеспечивая решение поисковых и аналитических задач [111].

В 2009 г. была представлена программно-аппаратная система научно-практического поиска и управления мульмедийным контентом (текстовым, графическим и аудиовизуальным) по массиву дел, образующихся в гражданских судах Испании. Система обеспечивала автоматическую классификацию изображений и сегментов аудиовизуальных записей в сочетании с текстовой семантикой и была ориентирована на получение новых знаний на основе процедурной онтологии e-Sentencias [17].

В 2016 г. была создана и в дальнейшем совершенствовалась система управления правовыми документами и знаниями EUNOMOS. На основе инструментов обработки естественного языка эта система в полуавтоматическом режиме обеспечивала классификацию юридических документов, выявляя перекрестные ссылки и законодательные изменения, связывая

юридические термины и извлекая ключевые элементы правовых норм для обеспечения ясности и расширенного поиска [13].

К 2017 г. в Великобритании была разработана система CLIEL, которая обеспечивает аннотирование юридических документов с использованием тегов XML для упрощения извлечения данных для разных типов точек данных. CLIEL представляет собой гибкую и масштабируемую среду извлечения данных из разнородных (по форматам, структурам и макетам) документов в предметной области коммерческого права с использованием методологии NLP и компонентов общей архитектуры текстовой инженерии (GATE), включая правила механизма шаблонов аннотаций Java (JAPE) [31].

4.8. Правовые информационные системы контрольного назначения

Развитие «электронного правительства» вызвало исследования в сфере контрольных интеллектуальных правовых систем. Так, в 2007 г. в качестве онтолингвистического ресурса была обоснована характеристика знаний DALOS и система организации знаний (KOS) для обеспечения контроля и поддержки законодательного процесса в многоязычной среде европейских стран, обязанных имплементировать евродирективы в национальное право [1].

В 2019 г. были представлены результаты разработки и апробации моделей на данных, обеспечивающих контроль над имплементацией европейского права в национальные правовые системы; эти модели были реализованы с использованием неконтролируемых методов лексического и семантического сходства, основанных на моделях векторного пространства, скрытом семантическом анализе и тематических моделях [88].

5. Методы машинного обучения и большие данные в предметной области права

Еще два исследовательских направления возникли уже в XXI веке и связаны с развитием методов машинного обучения и обработкой больших данных.

5.1. Методы машинного обучения в праве

Внедрение в юридическую область методов машинного обучения и извлечения данных осуществляется в самых разных областях права и для решения самых разнообразных задач.

Например, в 2005 г. в Эдинбургском университете Бен Хачи и Клэр Гровер в целях структурирования резюме документа провели эксперименты по построению риторической схемы аннотации в качестве модели правового дискурса, в которых на примере решений Палаты лордов прогнозировался риторический статус отдельных предложений в тексте и генерировалась ключевая фраза [41].

В 2005 г. словенско-британский коллектив предложил для поиска правовых обоснований использовать на массиве судебных дел алгоритмы машинного обучения ABCN2, являющиеся расширением индукционного алгоритма Кларка – Ниблетта и опирающиеся на некоторые концепции правовой аргументации [87].

В 2013 г. американские ученые использовали машинное обучение для извлечения и генерирования правовой аргументации по базе аннотированных судебных решений о компенсации вреда здоровью, причиненного вакинацией (V/IP Corpus), с применением интеграции семантико-прагматических, синтаксических и общедоменных семантических аннотаций [7]. В 2015 г. в США успешно использовались статистическое машинное обучение и прогностические модели для классификации нормативных текстов в специфических функциональных терминах на примере законодательства разных штатов об обеспечении готовности и реагировании системы здравоохранения в чрезвычайных ситуациях [109].

В 2019 г. на основе алгоритмов машинного обучения была разработана система CLAUDETTE, которая автоматически выявляет на платформах интернет-торговли

предложения, ущемляющие правв потребителя [70], а также были предложены методы контролируемого и неконтролируемого машинного обучения для автоматического тематического распределения судебных и любых других юридических документов [126].

5.2. Право и большие данные

Это направление, еще более новое, находится в стадии зарождения и состоит в изучении и обработке больших данных. Такие исследования тесно связаны с предыдущим направлением, поскольку извлечение информации на таких объемах данных часто связано с необходимостью использования методов машинного обучения.

«Большие данные» используются для исследований в самых разных сферах правовой деятельности. Так, несколько лет назад было предложено использование «больших данных» исторических источников для обоснования идей оригинализма в практике исторического толкования Конституции США [79]. А в Австралии «большие метаданные» Интернет-провайдеров используются в оперативно-разыскной деятельности, причем «большой» объем данных и использование алгоритмов для машинной обработки и интеллектуального анализа позиционируются в качестве определенной гарантии деиндентификации и анонимизации персональных данных и, как следствие, препятствия дискриминационному профилированию [75].

6. Современные отечественные достижения

Современные исследования и разработки на стыке права и компьютерных наук либо сконцентрированы в правоохранительной области, где применяются довольно широко, либо носят более универсальный характер, но в практической юриспруденции еще не заняли подобающее место.

6.1. Компьютерные системы и методы правоохранительного назначения

В России развивается компьютерное моделирование, ориентированное на правоохранительную сферу: на основе криминалистических знаний создаются компьютерные модели преступлений определенного вида в целях автоматизации методики их расследования [59], компьютерные модели внедряются в область судебных экспертиз [37, 64, 68, 74]. Известен опыт разработки компьютерных имитационных моделей в сфере профилактики коррупции [118].

Развивается направление компьютерных информационных систем, используемых в раскрытии и расследовании преступлений [21, 67, 132] и в экспертно-криминалистических целях [47, 89].

Отдельную область составляет внедрение технологий прикладного («слабого») искусственного интеллекта: в частности, в оперативно-разыскную деятельность [93] и криминалистику [8].

Однако в сфере правоохранительной деятельности в большей степени ведется внедрение уже известных типов информационных систем, и только в отдельных случаях [122] можно уверенно говорить о том, что представители компьютерных наук вырабатывают для правоохранительной области новые методологические решения.

6.2. Компьютерные методы в универсальных правовых исследованиях

В неспецифичных (регулятивных) областях правовой деятельности научный интерес к компьютерным системам и методам стал проявляться после успехов 2000-х гг. в области цифровизации публичного управления и государственных (муниципальных) услуг. В

последние годы российские исследования в данном сегменте включают следующие важнейшие направления.

Во-первых, ведутся работы в области предметно-ориентированных онтологий, которые предполагается использовать для создания компьютерных систем правотворческого и правоприменительного назначения, основанных на базах знаний. Одни исследования в этой области сосредоточены на экспериментировании с выразительностью онтологии [63], тогда как другие сопрягаются с методами NLP и интеллектуального анализа данных [83].

Во-вторых, проводятся эксперименты по применению методов обработки естественного языка на массивах юридических документов: например, объектных графов [131] и метрик частности терминов и обратной частотности документов (TF-IDF) [83].

В-третьих, на больших данных судебной практики проводятся вычислительные эксперименты по интеллектуальному анализу данных и машинному обучению, включая анализ временных рядов и регрессионные деревья [84, 85, 127].

7. Заключение

Систематический обзор позволяет сделать ряд выводов о результатах и состоянии исследований в сфере использования компьютерных методов и систем в изучении права, интеллектуальном анализе данных и моделировании правовой деятельности.

Во-первых, исследования в данном сегменте с конца 1950-х гг. ведутся достаточно активно и находятся в общем русле развития компьютерных наук, имплементируя в область права их новейшие достижения. Однако лишь незначительная доля достижений в области информатизации правовой деятельности основана на методологических прорывах междисциплинарного свойства. Чаще всего вычислительные системы и методы реализуют в праве уже известные и апробированные в других предметных областях решения, поскольку реализация научных проектов требует больших затрат ресурсов (времени, труда, материально-технических, вычислительных), а иногда также трансформации самой правовой деятельности в условиях неопределенной эффективности и адаптивности предлагаемых систем и методов.

Во-вторых, зачастую методологического успеха достигали исследовательские коллективы междисциплинарного состава, обеспечивавшие грамотное целеполагание, научную коммуникацию, моделирование объектов исследования и разработку средств реализации междисциплинарных подходов.

В-третьих, в отличие от большинства других областей общественной практики, правовая деятельность, включая ее языковое выражение, одновременно крайне формализована и крайне нестабильна, и это фундаментальное противоречие затрудняет автоматизацию операций с юридическими данными, документами, текстами, процессами и другими объектами.

В-четвертых, вычислительные методы и системы меняют правовую деятельность и повышают ее эффективность, но одновременно ставят ее в зависимость от технологий. В частности, не обладая абсолютной полнотой и точностью и используя сложную методологию, информационная система потенциально провоцирует излишнее к себе доверие, препятствует эксперту в понимании, выявлении и исправлении недостатков функционирования системы, коррелирует с низким профессиональным уровнем пользователей и затрудняет оспаривание и опровержение ошибочных выводов.

И, наконец, систематический обзор показывает, что практически все успешные методологические решения с конца 1950-х гг. сохраняют свое значение. Некоторые из них применяются непосредственно, другие стали основой для дальнейшего развития вычислительных методов и информационных систем в правовой деятельности.

Без понимания основных научных достижений и тенденций на стыке компьютерных и правовых наук вряд ли возможен существенный научный прогресс в этой сложной

междисциплинарной области, зато существует риск повторения уже известных ошибок и заблуждений.

Список литературы / References

[1] Agnoloni T., Bacci L., Francesconi E., Spinosa P., Tiscornia D., Montemagni S., Venturi G. Building an ontological support for multilingual legislative drafting. In Legal Knowledge and Information Systems (JURIX'2007), Amsterdam, IOS Press, 2007, pp. 9–18.

[2] Aikenhead M. Legal knowledge based systems: some observations on the future. Web Journal of Current Legal Issues, vol. 2, 1995, p. 72. Available at: http://www.bailii.org/uk/other/journals/WebJCLI/1995/issue2/aiken2.html.

[3] Al-Abdulkarim L., Atkinson K., Bench-Capon T. A methodology for designing systems to reason with legal cases using abstract dialectical frameworks. Artificial Intelligence and Law, vol. 24, no. 1, 2016, pp. 1–49.

[4] Андреев Н.Д., Керимов Д.А. О возможностях кибернетики при решении правовых проблем. Советское государство и право, № 7, 1960 г., стр. 106–110 / Andreev N.D., Kerimov D.A. Concerning the possibilities of cybernetics in solving legal problems. Soviet State and Law, no. 7, 1960, pp. 106–110 (in Russian).

[5] Апт Л.Ф., Цивилева Е.Д. «АИПС-законодательство» и ее проблемы (анализ результатов материалов поиска правовой информации). Проблемы совершенствования советского законодательства. Труды. М.: ВНИИСЗ, вып. 35, 1986 г., стр. 13–22 / Apt L.F., Tsivileva E.D. «AIRS-Legislation» and its problems (analysis of the results of legal information search materials). Problems of Improving Soviet Legislation. Proceedings, Moscow: All-Union Scientific Research Institute of Soviet Legislation, issue 35, 1986, pp. 13–22 (in Russian).

[6] Ashley K.D., Brüninghaus S. Automatically classifying case texts and predicting outcomes. Artificial Intelligence and Law, vol. 17, no. 2, 2009, pp. 125–165.

[7] Ashley K.D., Walker V.R. Toward constructing evidence-based legal arguments using legal decision documents and machine learning. In Proc. of the 14th International Conference on Artificial Intelligence and Law (ICAIL'13), 2013, pp. 176–180.

[8] Бахтеев Д.В. Искусственный интеллект в криминалистике: состояние и перспективы использования. Российское право: образование, практика, наука. № 2, 2018, с. 43–49. / Bakhteev D.V. Artificial intelligence in criminalistics: state and prospects of use. Russian Law: Education, Practice, Researches, no. 2, 2018, pp. 43–49 (in Russian).

[9] Baroni P., Giacomin M. Semantics of abstract argument systems. In Argumentation in artificial intelligence. Boston, Springer, 2009, pp. 25–44.

[10] Башев А.А., Москвин С.С., Фукс Н.С. Автоматизированная информационно-поисковая система по законодательству в области лесного хозяйства. Актуальные проблемы теории и практики применения математических методов и ЭВМ в деятельности органов юстиции. Тезисы докладов на V всесоюзной конференции по проблемам правовой кибернетики, вып. 2, 1975 г., стр. 28–34 / Bashev A.A., Moskvin S.S., Fuks N.S. Automated information retrieval system for forestry legislation. In Actual Problems of the Theory and Practice of the Application of Mathematical Methods and Computers in the Activities of the Judiciary, Abstracts at the 5th All-Union Conference on Legal Cybernetics, issue 2, 1975, pp. 28–34 (in Russian).

[11] Bench-Capon T.J.M. Before and after Dung: Argumentation in AI and law. Argument and Computation, vol. 11, no. 1, 2019, pp. 1–18.

[12] Bing J. Conceptual text retrieval. Oslo, Tano, 1988, 109 p.

[13] Boella G., Di Caro L., Leone V. Semi-automatic knowledge population in a legal document management system. Artificial Intelligence and Law, vol. 27, no. 2, 2019, pp. 227–251.

[14] Boucher J. Le projet Datum: Recherche sur un instrument de recherché. La Revue juridique Thémis, vol. 6, no. 1, 1971, pp. 31–49. (in French).

[15] Buchanan B.G., Headrick T.E. Some speculation about artificial intelligence and legal reasoning. Stanford Law Review, vol. 23, no. 1, 1970, pp. 40–62.

[16] Bundesministerium der Justiz (Hrsg.). Das Juristische Informationssystem. Analyse, Planung, Vorschläge. Karlsruhe: Verlag C. F. Müller, 1972. (in German).

[17] Casanovas P., Binefa i Valls X., Gracia C., Teodoro E., Galera N., Blázquez M., Poblet M., Carrabina J., Monton M., Montero C., Serrano J., López-Cobo J.M. The e-Sentencias prototype: A procedural ontology for legal multimedia applications in the Spanish civil courts. In Breuker J., Casanovas P., Klein M.C.A.,

Francesconi E. (eds.). Law, Ontologies and the Semantic Web: Channelling the Legal Information Flood, Amsterdam, IOS Press, 2009, pp. 199–219.

[18] Dadgostari F., Guim M., Beling P.A., Livermore M.A., Rockmore D.N. Modeling law search as prediction. Artificial Intelligence and Law, 2020, pp. 1–32. Available at: https://link.springer.com/article/10.1007%2Fs10506-020-09261-5.

[19] Давыденко В.К. Теоретические проблемы социально-правовой эффективности автоматизированных информационно-поисковых систем правовой информации. Автореф. дис. <…> канд. юрид. наук. М., 1980 г., 16 стр. / Davydenko V.K. Theoretical problems of social and legal effectiveness of automated information retrieval systems of legal information. Abstract of the dissertation of the candidate of legal sciences, Moscow, 1980 (in Russian).

[20] Деев А.Ф., Гальперин Л.Б., Иванов Ю.Т. Кибернетика и опыт решения некоторых правовых задач. Советское государство и право, № 10, 1964, с. 81–90. / Deev A.F., Gal'perin L.B., Ivanov Yu.T. Cybernetics and experience in solving some legal problems. Soviet State and Law, no. 10, 1964 г., pp. 81–90 (in Russian).

[21] Дубровин И.С. Информационно-поисковые системы отечественных, зарубежных и международных служб правоохранительных органов в борьбе с преступностью. Дис. <…> канд. юрид. наук. М., 2007 г., 192 стр. / Dubrovin I.S. Information retrieval systems of domestic, foreign and international law enforcement agencies in the fight against crime. Dissertation of the candidate of legal sciences, Moscow, 2007 (in Russian).

[22] Dung P.M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence, vol. 77, no. 2, 1995, pp. 321–357.

[23] Dung P.M., Son T.C. An argument-based approach to reasoning with specificity. Artificial Intelligence, vol. 133, no. 1–2, 2001, pp. 35–85.

[24] Dung P.M., Kowalski R.A., Toni F. Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence, vol. 170, no. 2, 2006, pp. 114–159.

[25] Dunn M., Sagun L., Şirin H., Chen D. Early predictability of asylum court decisions. In Proc. of the 16th International Conference on Artificial Intelligence and Law (ICAIL'17), 2017, pp. 233–236.

[26] Эджубов Л.Г. Использование некоторых методов и средств кибернетики в дактилоскопии. Дис. <…> канд. юрид. наук. М., 1962 г., 274 стр. / Ehdzhubov L.G. Using some methods and means of cybernetics in fingerprinting. Dissertation of the candidate of legal sciences, Moscow, 1962. (in Russian).

[27] Эджубов Л.Г., Литинский С.А. Результаты и перспективы применения ЭВМ в судебно-автотехнической экспертизе. Вопросы кибернетики, вып. 40, 1977 г., стр. 118–131 / Ehdzhubov L.G., Litinskii S.A. Results and prospects for the use of computers in criminalistics automotive expertise. In Cybernetics Issues, issue 40, 1977, pp. 118–131 (in Russian).

[28] Eldridge W.B. The American Bar Foundation project. MULL: Modern Uses of Logic in Law, vol. 6, no. 3, 1965, pp. 129–131.

[29] Гафинова И.Н., Литвинова Л.О., Москвин С.С., Трофимова И.В. Режим и порядок работы ИПС «Право-1». Правовая кибернетика. Сборник статей. М., Наука, 1973 г., стр. 43–54 / Gafinova I.N., Litvinova L.O., Moskvin S.S., Trofimova I.V. The mode and operating procedure of the ILS "Pravo-1". In Legal Cybernetics: A Collection of Articles, Moscow, Science, 1973, pp. 43–54 (in Russian).

[30] García R., Delgado J. An ontological approach for the management of Rights Data Dictionaries. In Legal Knowledge and Information Systems (JURIX'2005), Amsterdam, IOS Press, 2005, pp. 137–146.

[31] García-Constantino M., Atkinson K., Bollegala D., Chapman K., Coenen F., Roberts C., Robson K. CLIEL: Context-based information extraction from commercial law documents. In Proc. of the 16th International Conference on Artificial Intelligence and Law (ICAIL'17), 2017, pp. 79–87.

[32] Гаврилов О.А., Панкратов В.В., Эджубов Л.Г. Обсуждение использования методов статистики в юридической науке. Советское государство и право, № 10, 1966 г., стр. 159–160 / Gavrilov O.A., Pankratov V.V., Ehdzhubov L.G. Discussion of the application of statistical methods in legal science. Soviet State and Law, no. 10, 1966, pp. 159–160 (in Russian).

[33] Гегечкори Л.А., Шмелев А.А. Задачи совершенствования поискового аппарата автоматизированных информационно-поисковых систем по законодательству. Правовая информатика. Сборник. М., вып. 1, 1996 г., стр. 63–69 / Gegechkori L.A., Shmelev A.A. The objectives of improving the search for automated legal retrieval systems. In Legal Informatics: A Collection, Moscow, issue 1, 1996, pp. 63–69 (in Russian).

[34] Gelbart D., Smith J.C. FLEXICON: An evaluation of a statistical ranking model adapted to intelligent legal text management. In Proc. of the 4th International Conference on Artificial Intelligence and Law (ICAIL'93), 1993, pp. 142–151.

[35] Gifford M. LexrideLaw: An argument based legal search engine. In Proceedings of the 16th International Conference on Artificial Intelligence and Law (ICAIL'17), 2017, pp. 271–272.

[36] Гиринский В.Е., Журавлев В.А., Кулешов В.П., Резников В.Б. О применении ЭВМ для учета и обработки данных о лицах, задержанных за бродяжничество и попрошайничество. Применение математических методов и вычислительной техники в праве, криминалистике и судебной экспертизе. Материалы симпозиума. М., 1970 г., стр. 80–81 / Girinskii V.E., Zhuravlev V.A., Kuleshov V.P., Reznikov V.B. On the use of computers for recording and processing data on persons detained for vagrancy and begging. In Application of Mathematical Methods and Computer Technology in Law, Criminalistics and Criminalistics Expertise: Proceedings of the Symposium, Moscow, 1970, pp. 80–81 (in Russian).

[37] Головчанский А.В. Применение компьютерного моделирования при установлении обстоятельств дорожно-транспортных происшествий. Обеспечение прав и законных интересов граждан в деятельности органов дознания и предварительного следствия. Сборник статей. Орел, ОрЮИ МВД России имени В. В. Лукьянова, 2018 г., стр. 54–64. / Golovchanskii A.V. The use of computer modeling to establish the circumstances of traffic accidents. In Ensuring the Rights and Legitimate Interests of Citizens in the Activities of Bodies of Inquiry and Investigation: A Collection of Articles, Oryol, Oryol Law Institute of the Ministry of Internal Affairs of Russia, 2018, pp. 54–64 (in Russian).

[38] Grabmair M. Predicting trade secret case outcomes using argument schemes and learned quantitative value effect tradeoffs. In Proc. of the 16th International Conference on Artificial Intelligence and Law (ICAIL'17), 2017, pp. 89–98.

[39] Грановский Г.Л., Пименов Н.Ф., Эджубов Л.Г. Использование математических методов и электронно-вычислительных машин в трасологической экспертизе. Проблемы и практика трасологических и баллистических исследований. М., ВНИИСЭ, 1976 г., стр. 25–42 / Granovskii G.L., Pimenov N.F., Ehdzhubov L.G. The use of mathematical methods and electronic computers in trasological examination. In Problems and Practice of Trasological and Ballistic Research, Moscow, All-Union Research Institute for Criminalistics Expert Examination, 1976, pp. 25–42 (in Russian).

[40] Griffo C., Almeida J.P.A., Guizzardi G. A pattern for the representation of legal relations in a legal core ontology. In Legal Knowledge and Information Systems (JURIX'2016), Amsterdam, IOS Press, 2016, pp. 191–194.

[41] Hachey B., Grover C. Automatic legal text summarization: experiments with summary structuring. In Proc. of the 10th International Conference on Artificial Intelligence and Law (ICAIL'2005), 2005, pp. 75–84.

[42] Hafner C.D. Representation of knowledge in a legal information retrieval system. In Proc. of the 3rd Annual ACM Conference on Research and Development in Information Retrieval (SIGIR'80), 1981, pp. 139–153.

[43] Harrington W.G., Wilson H.D., Bennett R.N. The Mead Data Central system of computerized legal research. Law Library Journal, vol. 64, no. 2, 1971, pp. 185–189.

[44] Hoppenfeld E.C. Law Research Service/Inc. MULL: Modern Uses of Logic in Law, vol. 7, no. 1, 1966, pp. 46–52.

[45] Horty J.F. Experience with the application of electronic data processing systems in general law. MULL: Modern Uses of Logic in Law, vol. 2, no. 4, 1960, pp. 158–168.

[46] Horty J.F. The «key words in combination» approach. MULL: Modern Uses of Logic in Law, vol. 3, no. 1, 1962, pp. 54–64.

[47] Каримов В.Х. Автоматизированные информационно-поисковые системы криминалистического назначения: современное состояние, тенденции и перспективы развития. М., Юрлитинформ, 2014 г.. 151 стр. / Karimov V.Kh. Automated criminalistics information retrieval systems: current status, trends and development prospects, Moscow, Yurlitinform, 2014 (in Russian).

[48] Каск Л.И. О некоторых вопросах информационного языка для права. Вестник Ленинградского университета. Серия экономики, философия и права, вып. 2, № 11, 1961 г., стр. 135–138. / Kask L.I. Concerning some issues of information language for law. Bulletin of the Leningrad University. Series: Economics, Philosophy and Law, issue 2, no. 11, 1961, pp. 135–138 (in Russian).

[49] Kehl W.B., Horty J.F., Bacon C.R.T., Mitchell D.S. An information retrieval language for legal studies. Communications of the ACM, vol. 4, no. 9, 1961, pp. 380–389.

[50] Керимов Д.А. Об использовании кибернетических машин в процессе кодификации советского права. Вопросы кодификации советского права. Л., Издательство Ленинградского. университета, вып. 3, 1960 г., стр. 121–123. / Kerimov D.A. On the use of cybernetic machines in the codification of Soviet law. In Issues of Codification of Soviet Law, Leningrad, Leningrad University, issue 3, 1960, pp. 121–123 (in Russian).

[51] Kerimov D.A. Future applicability of cybernetics to jurisprudence in the USSR. MULL: Modern Uses of Logic in Law, vol. 4, no. 4, 1963, pp. 153–162.

[52] Керимов Д.А. Право и кибернетика. Советское государство и право, № 9, 1964 г., стр. 86–94 / Kerimov D.A. Law and cybernetics. Soviet State and Law, no. 9, 1964, pp. 86–94 (in Russian).

[53] Керимов Д.А. О справочно-информационной службе в области права. Вопросы кибернетики и право. Сборник статей. М., Наука, 1967 г., стр. 61–83 / Kerimov D.A. Concerning the legal reference service. In Cybernetics and Law: A Collection of Articles, Moscow, Science, 1967, pp. 61–83 (in Russian).

[54] Керимов Д.А., Эджубов Л.Г. Как возникла правовая кибернетика. Путь в большую науку: академик Аксель Берг. Сборник статей. М., Наука, 1988 г., стр. 234–243 / Kerimov D.A., Ehdzhubov L.G. The rise of legal cybernetics. In The Path to Big Science: Academician Axel Berg. A Collection of Articles, Moscow, Science, 1988, pp. 234–243 (in Russian).

[55] Керимов Д.А., Покровский И.Ф. Опыт использования средств кибернетики для автоматизации информационной службы в области права. Вестник Ленинградского университета. Серия экономики, философии и права, вып. 1, № 5, 1964 г., стр. 121–124 / Kerimov D.A., Pokrovskii I.F. Experience in using cybernetics to automate information services in the field of law. Bulletin of the Leningrad University. Series: Economics, Philosophy and Law, issue 1, no. 5, 1964, pp. 121–124 (in Russian).

[56] Хвыля-Олинтер А.И. Использование криминалистической характеристики преступлений в автоматизированных информационно-поисковых системах технико-криминалистического назначения: дис. <…> канд. юрид. наук. М., 1995 г., 225 стр. / Khvylya-Olinter A.I. Using the criminalistics characteristics of crimes in automated information retrieval systems for technical and criminalistics purposes: dissertation of the candidate of legal sciences, Moscow, 1995 (in Russian).

[57] Кирюшкин М.В. Алгоритмические преобразования в юриспруденции. Российский юридический журнал, № 4, 2007 г., стр. 34–44 / Kiryushkin M.V. Algorithmic transformations in jurisprudence. Russian Juridical Journal, no. 4, 2007, pp. 34–44 (in Russian).

[58] Копылов В.А. Методы комплексного создания и применения динамических автоматизированных информационных систем для обработки слабоформализуемой информации: автореф. дис. <…> д-ра техн. наук. М., 1994 г., 67 стр. / Kopylov V.A. Methods for the integrated creation and use of dynamic automated information systems for processing semi-formalized information: abstract of the dissertation of the doctor of technical sciences, Moscow, 1994 (in Russian).

[59] Ковалев С.А. Основы компьютерного моделирования при расследовании преступлений в сфере компьютерной информации: дис. <…> канд. юрид. наук. Волгоград, 2012 г., 259 стр. / Kovalev S.A. Fundamentals of computer modeling in the investigation of crimes in the field of computer information: dissertation of the candidate of legal sciences, Volgograd, 2012 (in Russian).

[60] Козинец Б.Н., Ланцман Р.М., Якубович В.А. Криминалистическая экспертиза близких почерков при помощи электронно-вычислительных машин. Доклады АН СССР, том 167, № 5, 1966, с. 1008–1011 / Kozinets B.N., Lantsman R.M., Yakubovich V.A. Criminalistics expert examination of close handwriting using electronic computers. Proceedings of the USSR Academy of Sciences, vol. 167, no. 5, 1966, pp. 1008–1011 (in Russian).

[61] Козинец Б.Н., Ланцман Р.М., Якубович В.А. Об одном кибернетическом методе исследования в криминалистической экспертизе почерка. Кибернетика и судебная экспертиза. Вильнюс, Изд-во НИИСЭ, 1966 г, стр. 55–84 / Kozinets B.N., Lantsman R.M., Yakubovich V.A. Concerning one cybernetic research method in the criminalistics expert examination of handwriting. In Cybernetics and Criminalistics Expertise, Vilnius, Research Institute for Criminalistics Expert Examination, 1966, pp. 55–84 (in Russian).

[62] Кудрявцев В.Н. О программировании процесса применения норм права. Вопросы кибернетики и право. Сборник статей. М., Наука, 1967 г., стр. 84–99. / Kudryavtsev V.N. Concerning programming the process of applying the rule of law. In Cybernetics and Law: A Collection of Articles, Moscow, Science, 1967, pp. 84–99 (in Russian).

[63] Kurcheeva G., Rakhvalova M., Rakhvalova D., Bakaev M. Mining and indexing of legal natural language texts with domain and task ontology. In Proc. of the 5th International Conference on Electronic Governance and Open Society: Challenges in Eurasia (EGOSE'2018), 2019, pp. 123–137.

[64] Ларионова Е.Ю., Голодков Ю.Э., Баранов С.А. Использование компьютерного моделирования для установления структуры химических соединений – объектов судебных экспертиз. Деятельность правоохранительных органов в современных условиях. Материалы XVIII международной научно-практической конференции, посвященной 20-летию образования института. Иркутск, Вост.-Сибир. ин-т МВД России, 2013 г, стр. 265–269. / Larionova E.Yu., Golodkov Yu.Eh., Baranov S.A. Using

computer modeling to establish the structure of chemical compounds – objects of criminalistics expertise. In The activities of law enforcement agencies in modern conditions: Proc. of the 18th International Scientific and Practical Conference Dedicated to the 20th Anniversary of the Institute, Irkutsk, East Siberian Institute of the Ministry of Internal Affairs of Russia, 2013, pp. 265–269 (in Russian).

[65] Lawford H. QUIC/LAW: Project of Queens' University. In Automated Law Research: A Collection of Presentations Delivered at the 1st National Conference on Automated Law Research. Chicago: ABA, 1973, pp. 67–93.

[66] Leimdörfer M. IMDOC. A computerized information retrieval system used by two governments in Europe. Law and Computer Technology, vol. 6, 1973, pp. 75–77.

[67] Леонов И.Н. Использование автоматизированных информационно-поисковых систем в раскрытии и расследовании преступлений, совершенных с применением огнестрельного оружия: дис. <…> канд. юрид. наук. М., 2006 г.. 172 стр. / Leonov I.N. The use of automated information retrieval systems in the detection and investigation of crimes committed with the use of firearms: dissertation of the candidate of legal sciences, Moscow, 2006 (in Russian).

[68] Лепихова Д.Н., Гудков В.Ю., Кирсанова А.А. Обзор современных моделей представления дактилоскопических изображений. Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика, том 7, № 1, 2018 г., стр. 40–59 / Lepikhova D.N., Gudkov V.Yu., Kirsanova A.A. An overview of fingerprint description models. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering, vol. 7, no. 1, 2018, pp. 40–59 (in Russian).

[69] Libal T., Steen A. NAI – the normative reasoner. In Proc. of the 17th International Conference on Artificial Intelligence and Law (ICAIL'19), 2019, pp. 262–263.

[70] Lippi M., Pałka P., Contissa G., Lagioia F., Micklitz H.-W., Sartor G., Torroni P. CLAUDETTE: An automated detector of potentially unfair clauses in online terms of service. Artificial Intelligence and Law, vol. 27, no. 2, 2019, pp. 117–139.

[71] Loevinger L. Jurimetrics. The next step forward. Minnesota Law Review, vol. 33, no. 5, 1949, pp. 455–493.

[72] Losano M.G. Giuscibernetica. Macchine e modelli cibernetici nel diritto. Torino, G. Einaudi, 1969 (in Italian).

[73] Mackaay E. La création d'un thésaurus bilingue pour DATUM. La Revue juridique Thémis, vol. 6, no. 1, 1971, pp. 51–67 (in French).

[74] Макаров И.Ю., Светлаков А.В., Сотин А.В., Шигеев С.В., Гусаров А.А., Смиренин С.А., Емелин В.В., Страгис В.Б., Фетисов В.А. Эффективность использования современных компьютерных технологий в клинической практике и перспективы применения биомеханических 3D-моделей в судебной медицине. Судебно-медицинская экспертиза, том 61, № 2, 2018 г, стр. 58–64. / Makarov I.Yu., Svetlakov A.V., Sotin A.V., Shigeev S.V., Gusarov A.A., Smirenin S.A., Emelin V.V., Stragis V.B., Fetisov V.A. The efficiency of the application of the modern computed technologies in the clinical practice and the prospects for the further use of the biomechanical 3D-models in forensic medicine. Forensic Medical Expertise, vol. 61, no. 2, 2018, pp. 58–64 (in Russian).

[75] Maurushat A., Moses L.B., Vaile D. Using "big" metadata for criminal intelligence: Understanding limitations and appropriate safeguards. In Proc. of the 15th International Conference on Artificial Intelligence and Law (ICAIL'15), 2015, pp. 196–200.

[76] McCarty L.T. Reflections on Taxman: An experiment in artificial intelligence and legal reasoning. Harvard Law Review, vol. 90, no. 5, 1977, pp. 837–893.

[77] McCarty L.T. Intelligent legal information systems: Problems and prospects. Rutgers Computer and Technology Law Journal, vol. 9, no. 2, 1983, pp. 265–294.

[78] McCarty L.T., Sridharan N.S. The representation of an evolving system of legal concepts: II. Prototypes and deformations. In Proc. of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81), 1981, vol. I, pp. 246–253.

[79] McGinnis J.O., Stein B. Originalism, hypothesis testing and big data. In Proc. of the 15th International Conference on Artificial Intelligence and Law (ICAIL'15), 2015, pp. 201–205.

[80] Mehl L. Automation in the legal world: From the machine processing of legal information to the «law machine». In Mechanization of Thought Processes: Proc. of a Symposium Held at the National Physical Laboratory: National Physical Laboratory Symposium № 10, London: H. M. Stationery Office, 1959, vol. II, pp. 758–759.

[81] Melton J.S. The «semantic coded abstract» approach. MULL: Modern Uses of Logic in Law, vol. 3, no. 1, 1962, pp. 48–54.

[82] Melton J.S., Bensing R.C. Searching legal literature electronically: Results of a test program. Minnesota Law Review, vol. 45, no. 2, 1960, pp. 229–248.

[83] Metsker O., Trofimov E., Grechishcheva S. Natural language processing of Russian court decisions for digital indicators mapping for oversight process control efficiency: Disobeying a police officer case. In Proc. of the 6th International Conference on Electronic Governance and Open Society: Challenges in Eurasia (EGOSE'2019), 2020, pp. 295–307.

[84] Metsker O., Trofimov E., Petrov M., Butakov N. Russian court decisions data analysis using distributed computing and machine learning to improve lawmaking and law enforcement. In Proc. of the 8th International Young Scientist Conference on Computational Science (YSC'2019), 2019, pp. 264–273.

[85] Metsker O., Trofimov E., Sikorsky S., Kovalchuk S. Text and data mining techniques in judgment open data analysis for administrative practice control. In Proc. of the 5th International Conference on Electronic Governance and Open Society: Challenges in Eurasia (EGOSE'2018), 2019, pp. 169–180.

[86] Morgan R.T. The «point of law» approach. MULL: Modern Uses of Logic in Law, vol. 3, no. 1, 1962, pp. 44–48.

[87] Mozina M., Zabkar J., Bench-Capon T., Bratko I. Argument based machine learning applied to law. Artificial Intelligence and Law, vol. 13, no. 1, 2005, pp. 53–73.

[88] Nanda R., Siragusa G., Caro L.D., Boella G., Grossio L., Gerbaudo M., Costamagna F. Unsupervised and supervised text similarity systems for automated identification of national implementing measures of European directives. Artificial Intelligence and Law, vol. 27, no. 2, 2019, pp. 199–225.

[89] Немчин Д.И. Методические основы применения информационных компьютерных технологий в судебно-баллистической экспертизе: дис. <…> канд. юрид. наук. М., 2002 г., 161 стр. / Nemchin D.I. Methodological foundations of the use of information computer technologies in ballistic examination: dissertation of the candidate of legal sciences, Moscow, 2002 (in Russian).

[90] Niblett G.B.F., Price N.H. Mechanized searching of acts of Parliament. Information Storage and Retrieval, vol. 6, no. 3, 1970, pp. 289–297.

[91] Ножов И.М. Графематический и морфологический модули для решения задач автоиндексации текстов. Правовая информатика. Сборник. М., НЦПИ, 2001 г., стр. 49–55 / Nozhov I.M. Graphematical and morphological modules for solving problems of text auto-indexing. In Legal Informatics: A Collection, Moscow, Scientific Center of Legal Information, 2001, pp. 49–55 (in Russian).

[92] Олейников В.Т. Автоматизация процесса поиска в портретных учетах по измерительным признакам внешности. Вопросы совершенствования деятельности органов внутренних дел, Тезисы выступлений на I межвузовской научно-практической конференции адъюнктов и соискателей во ВНИИ МВД СССР. М., 1978 г., стр. 153–155. / Oleinikov V.T. Automation of the search process in portrait accounts by measuring signs of appearance. In Issues of Improving the Activities of the Internal Affairs Bodies. Abstracts of Speeches at the 1st Interuniversity Scientific and Practical Conference of Adjuncts and Applicants at the All-Union Scientific Research Institute of the USSR Ministry of Internal Affairs, Moscow, 1978, pp. 153–155 (in Russian).

[93] Овчинский А.С. Оперативно-разыскная аналитика на пути к искусственному интеллекту. Актуальные проблемы теории оперативно-розыскной деятельности. Сборник научных трудов. М., Инфра-М, 2017 г., стр. 364–393 / Ovchinskii A.S. Operational investigative analytics on the way to artificial intelligence. In Actual Problems of the Theory of Operational Investigative Activity: A Collection of Scientific Papers, Moscow, Infra-M, 2017, pp. 364–393 (in Russian).

[94] Парфентьев А.Л. Проблемы создания комплекса алгоритмов и программ отображения и оценки смысла юридических установлений в правовой автоматизированной информационной системе. Правовая информатика. Сборник. М., ЦНПИ, вып. 3, 1998 г., стр. 111–124 / Parfent'ev A.L. Problems of creating a complex of algorithms and programs for displaying and evaluating the meaning of legal institutions in a legal automated information system. In Legal Informatics: A Collection, Moscow, Scientific Center of Legal Information, 1998, issue 3, pp. 111–124 (in Russian).

[95] Полевой Н.С., Шляхов А.Р., Эджубов Л.Г. Использование кибернетики и математических методов в судебной экспертизе. Правоведение, № 6, 1972 г., стр. 124–131 / Polevoi N.S., Shlyakhov A.R., Ehdzhubov L.G. The use of cybernetics and mathematical methods in criminalistics expertise. Jurisprudence, no. 6, 1972, pp. 124–131 (in Russian).

[96] Popple J. A pragmatic legal expert system. Aldershot, Dartmouth, 1996, 406 p.

[97] Prakken H. Reconstructing Popov v. Hayashi in a framework for argumentation with structured arguments and Dungean semantics. Artificial Intelligence and Law, vol. 20, no. 1, 2012, pp. 57–82.

[98] Prakken H., Sartor G. A dialectical model of assessing conflicting arguments in legal reasoning. Artificial Intelligence and Law, vol. 4, no. 3–4, 1996, pp. 331–336.

[99] Prestel B.M. CREDOC: Centre de documentation juridique Bruxelles. In Materialien zur Rechtsinformatik. Folg 1: Länderberichte USA, Schweden; Dokumentationssysteme CREDOC, UNIDATA; Bibliographie. Frankfurt am Main: Alfred Metzner Verlag, 1971, s. 55–70 (in German).

[100] Рашитов Р.С. Система автоматической дактилоскопической регистрации на базе специализированной ЭВМ «Минск-100». Проблемы правовой кибернетики. Материалы симпозиума. М., 1968 г., стр. 218–220. / Rashitov R.S. Automatic fingerprint registration system based on specialized computer «Minsk-100». In Problems of Legal Cybernetics: Symposium Proceedings, Moscow, 1968, pp. 218–220 (in Russian).

[101] Rissland E.L., Ashley K.D. HYPO: A precedent-based legal reasoner. Amherst, University of Massachusetts, 1987, 25 p.

[102] Rissland E.L., Skalak D.B. CABARET: Rule interpretation in a hybrid architecture. International Journal of Man-Machine Studies, vol. 34, no. 6, 1991, pp. 839–887.

[103] Rissland E.L., Daniels J.J., Rubinstein Z.B., Skalak D.B. Case-based diagnostic analysis in a blackboard architecture. In Proc. of the 11th National Conference on Artificial Intelligence (AAAI'93), 1993, pp. 66–72.

[104] Rissland E.L., Skalak D.B., Friedman M.T. BankXX: Supporting legal arguments through heuristic retrieval. Artificial Intelligence and Law, vol. 4, no. 1, 1996, pp. 1–71.

[105] Rissland E.L., Skalak D.B., Friedman M.T. Evaluating a legal argument program: The BankXX experiments. Artificial Intelligence and Law, vol. 5, no. 1–2, 1997, pp. 1–74.

[106] Rubin J.S. LEXIS has made computer-assisted legal research in the United States a practical reality. Law and Computer Technology, vol. 7, no. 2, 1974, pp. 34–50.

[107] Sanders K.E. CHIRON: Planning in an open-textured domain. Artificial Intelligence and Law, vol. 9, no. 4, 2001, pp. 225–269.

[108] Saravanan M., Ravindran B., Raman S. Using legal ontology for query enhancement in generating a document summary. In Proc. of the Twentieth Annual Conference on Legal Knowledge and Information Systems (JURIX 2007), 2007, pp. 171–172.

[109] Savelka J., Ashley K.D. Transfer of predictive models for classification of statutory texts in multijurisdictional settings. In Proc. of the 15th International Conference on Artificial Intelligence and Law (ICAIL'15), 2015, pp. 216–226.

[110] Schank R.C., Kolodner J.L., DeJong G. Conceptual information retrieval. In Proc. of the 3rd Annual ACM Conference on Research and Development in Information Retrieval (SIGIR'80), 1981, pp. 94–116.

[111] Schweighofer E., Winiwarter W. Legal expert system KONTERM – Automatic representation of document structure and contents. In Database and Expert Systems Applications: 4th International Conference (DEXA'93), 1993, pp. 486–497.

[112] Шахтарина Н.И. Применение вероятностно-статистических методов оценки в судебно-почерковедческой экспертизе. Проблемы правовой кибернетики. Материалы симпозиума. М., 1968 г., стр. 181–184 / Shakhtarina N.I. The use of probabilistic and statistical assessment methods in criminalistics handwriting examination. In Problems of Legal Cybernetics: Symposium Proceedings, Moscow, 1968, pp. 181–184 (in Russian).

[113] Щербинин А.И., Юрьев В.П. Опыт разработки, внедрения и эксплуатации автоматизированных информационно-поисковых систем по учету автотранспортных средств в ГАИ. Пособие. М., изд-во ВНИИ БД МВД СССР, 1978 г., 71 стр. / Shcherbinin A.I., Yur'ev V.P. Experience in the development, implementation and operation of automated information retrieval systems for the registration of vehicles in the traffic police: a manual, Moscow: All-Union Research Institute for Traffic Safety of the Ministry of Internal Affairs of the USSR, 1978, 71 p. (in Russian).

[114] Шляхов А.Р., Эджубов Л.Г. Современное состояние и некоторые проблемы использования кибернетики в праве. Советское государство и право, no. 6, 1965 г., стр. 83–92 / Shlyakhov A.R., Ehdzhubov L.G. Current status and some problems of the use of cybernetics in law. Soviet State and Law, no. 6, 1965, pp. 83–92 (in Russian).

[115] Stevens C., Barot V., Carter J. The next generation of legal expert systems – New dawn or false dawn? In Proc. of the International Conference on Innovative Techniques and Applications of Artificial Intelligence, 2011, pp. 439–452.

[116] Stranieri A., Zeleznikow J. The SPLIT-UP System: Integrating neural networks and rule-based reasoning in the legal domain. In Proc. of the 5th International Conference on Artificial Intelligence and Law (ICAIL'95), 1995, pp. 185–194.

[117] Stranieri A., Zeleznikow J., Gawler M., Lewis B. A hybrid-neural approach to the automation of legal reasoning in the discretionary domain of family law in Australia. Artificial Intelligence and Law, vol. 7, no. 2–3, 1999, pp. 153–183.

[118] Судохолов А.П., Кузнецова И.А. Коррупция: механизмы развития, способы профилактики (опыт компьютерного моделирования с применением численных методов). Вестник Российского университета дружбы народов. Серия: Математика, информатика, физика, том 26, no. 2, 2018 г., стр. 183–193 / Sudokholov A.P., Kuznetsova I.A. Corruption: development mechanisms, ways of prevention (experience of computer modeling with application of numerical methods). RUDN Journal of Mathematics, Information Science and Physics Series, vol. 26, no. 2, 2018, pp. 183–193 (in Russian).

[119] Tapper C. Feasibility study of the retrieval of legal information from two types of natural language text. Research report № 5062, Oak Ridge: Office for Scientific and Technical Information, 1969.

[120] Tapper C. Legal information retrieval by computer: Applications and implications. McGill Law Journal, vol. 20, no. 1, 1974, pp. 26–43.

[121] Tapper C.F. British experience in legal information retrieval. MULL: Modern Uses of Logic in Law, vol. 5, no. 4, 1964, pp. 127–134.

[122] Ткаченко К.И. Автоматизированная информационная система формирования фактографических данных и ее применение для криминалистики, инновации и обучения: дис. <…> канд. техн. наук. М., 2017 г., 263 стр. / Tkachenko K.I. Automated information system for the formation of factual data and its application for criminalistics, innovation and training: dissertation of the candidate of technical sciences, Moscow, 2017 (in Russian).

[123] Tong R.M., Appelbaum L.A. Conceptual information retrieval using RUBRIC. In Proc. of the 10th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1987, pp. 247–253.

[124] Tong R.M., Appelbaum L.A. Experiments with interval-valued uncertainty. In Proc. of the Second Annual Conference on Uncertainty in Artificial Intelligence, 1988, pp. 63–75.

[125] Tong R.M., Aksman V.N., Cunningham J.F., Tollander C.J. RUBRIC: An environment for full text information retrieval. In Proc. of the 8th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1985, pp. 243–251.

[126] Torrisi A., Bevan R., Atkinson K., Bollegala D., Coenen F. Automated bundle pagination using machine learning. In Proc. of the 17th International Conference on Artificial Intelligence and Law (ICAIL'19), New York: ACM, 2019, pp. 244–248.

[127] Трофимов Е.В., Мецкер О.Г. Право и искусственный интеллект: опыт разработки вычислительной методологии для анализа и оценки качественных изменений в законодательстве и правоприменительной практике (на примере статьи 20.4 Кодекса Российской Федерации об административных правонарушениях). Право и политика, no. 8, 2019 г., стр. 1–17 / Trofimov E.V., Metsker O.G. Law and artificial intelligence: the experience of computational methodology for analyzing and assessing quantitative changes in legislation and law enforcement practice (on the example of the article 20.4 of the Code of the Russian Federation on Administrative Offenses). Law and Politics, no. 8, 2019, pp. 1–17 (in Russian).

[128] Troy F.J. Ohio Bar Automated Research – A practical system of computerized legal research. Jurimetrics Journal, vol. 10, no. 2, 1969, pp. 62–69.

[129] Трусов А.И. Судебное доказывание в свете идей кибернетики. Вопросы кибернетики и право. М., Наука, 1967 г., стр. 20–35 / Trusov A.I. Judicial proof in the light of cybernetics. In Cybernetics and Law Issues, Moscow, Science, 1967, pp. 20–35 (in Russian).

[130] Valente A., Breuker J. Making ends meet: Conceptual models and ontologies in legal problem solving. In Proc. of the XI Brazilian Symposium on Artificial Intelligence (SBIA'94), 1994, pp. 395–410.

[131] Васильев В.В., Грачева А.В., Родионов А.И., Блеканов И.С. Графовые методы выявления семантически значимых текстов судебных решений. Процессы управления и устойчивость, т. 6, no. 1, 2019 г., стр. 234–239 / Vasil'ev V.V., Gracheva A.V., Rodionov A.I., Blekanov I.S. Graph methods for identifying semantically significant texts of court decisions. Control Processes and Stability, vol. 6, no. 1, 2019, pp. 234–239 (in Russian).

[132] Вехов В.Б. Применение информационных систем специального назначения в раскрытии и расследовании преступлений. Оперативно-разыскное право. Сборник статей. Волгоград, Волгоград. акад. МВД России, 2013 г., стр. 26–32. / Vekhov V.B. The application of special-purpose information systems in the disclosure and investigation of crimes. In Operational Investigative Law: Collection of Articles, Volgograd, Volgograd Academy of the Ministry of Internal Affairs of Russia, 2013, pp. 26–32 (in Russian).

[133] Ветров А.Г. Основные проектные решения АИПС «Законодательство» – первой отечественной справочной правовой системы. Правовой мониторинг. М., ФГУ НПЦИ при Минюсте России, вып. 11, 2010 г., стр. 12–17 / Vetrov A.G. The main design decisions «AIRS-Legislation» – the first domestic reference legal system. In Legal Monitoring, Moscow: Scientific Center of Legal Information under the Ministry of Justice of Russia, 2010, issue 11, pp. 12–17 (in Russian).

[134] Вул С.М. Статистическое исследование текстов с помощью ЭВМ и дисплея в целях установления авторства. Применение ЭВМ в судебно-экспертных исследованиях и поиск правовой информации. М., ВНИИСЭ, 1975 г., стр. 227–233 / Vul S.M. Statistical study of texts using computers and display in order to establish authorship. In The Use of Computers in Criminalistics Expert Examination and the Search for Legal Information, Moscow, All-Union Research Institute for Criminalistics Expert Examination, 1975, pp. 227–233 (in Russian).

[135] Walker R.F., Oskamp A., Schrickx J.A., Opdorp G.J., van den Berg P.H. PROLEXS: Creating law and order in a heterogeneous domain. International Journal of Man-Machine Studies, vol. 35, no. 1, 1991, pp. 35–68.

[136] Wegstein J.H. A computer oriented single-fingerprint identification system. Technical Note 443; National Bureau of Standards, U.S. Department of Commerce, 1969.

[137] Юсупов С.Н. Информационно-поисковый язык по законодательству: автореф. дис. <…> канд. юрид. наук. М., 1974 г.. 20 стр. / Yusupov S.N. Legislative information retrieval language: abstract of the dissertation of the candidate of legal sciences, Moscow, 1974. (in Russian).

[138] Zeng Y., Wang R., Zeleznikow J., Kemp E. A knowledge representation model for the intelligent retrieval of legal cases. International Journal of Law and Information Technology, vol. 15, no. 3, 2007, pp. 299–319.

[139] Zhong L., Zhong Z., Zhao Z., Wang S., Ashley K.D., Grabmair M. Automatic summarization of legal decisions using iterative masking of predictive sentences. In Proc. of the 17th International Conference on Artificial Intelligence and Law (ICAIL'19), 2019, pp. 163–172.

[140] Зинин А.М., Крымский Н.К., Снетков В.А., Файн В.С. Исследование возможностей портретной идентификации с использованием средств электронно-вычислительной техники. Применение математических методов и вычислительной техники в праве, криминалистике и судебной экспертизе: материалы симпозиума. М., 1970 г., стр. 158–160 / Zinin A.M., Krymskii N.K., Snetkov V.A., Fain V.S. The study of the possibilities of portrait identification using electronic computers. In The Application of Mathematical Methods and Computer Technology in Law, Criminalistics and Criminalistics Expertise: Proceedings of the Symposium, Moscow, 1970, pp. 158–160 (in Russian).

Информация об авторах / Information about authors

Егор Викторович ТРОФИМОВ – доктор юридических наук, доцент, заслуженный юрист Республики Алтай, заместитель директора по научной работе. Область интересов: междисциплинарная компьютерно-юридическая методология, противодействие коррупции.

Egor Viktorovich TROFIMOV – Doctor of Law, associate professor, Honored Lawyer of the Altai Republic, deputy director for research. Research interests: interdisciplinary computer-legal methodology, anti-corruption.

Олег Геннадьевич МЕЦКЕР – кандидат технических наук, руководитель группы моделирования и прогнозирования управления по реализации федеральных проектов. Область интересов: интеллектуальные информационные системы, моделирование на данных, моделирование комплексных процессов, машинное обучение, прикладной искусственный интеллект, вычислительное здравоохранение, организация здравоохранения, административное право, юридические науки, обработка текстов на естественном языке, базы знаний, высокопроизводительные вычисления, человеко-машинное взаимодействие.

Oleg Gennad`evich METSKER – Candidate of Technical Sciences, head of the modeling and forecasting group of the department for the implementation of federal. Field of Interest: intelligent information systems, data modeling, data mining, complexity, uncertainty, machine learning, artificial intelligence, computational healthcare, legal science, text mining, process mining, natural language processing, knowledge bases, high-performance computing, big data, administrative sciences, e-sciences, human-computer interaction.