ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 34 Выпуск 5

ISSN Online 2220-6426
Volume 34 Issue 5

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2022

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:

УДК004.45

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

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2022-34(5)-1

Сравнение инструментов высокоуровневого синтеза и конструирования цифровой аппаратуры

[1,2,3,4,5] *А.С. Камкин, ORCID: 0000-0001-6374-8575 <kamkin@ispras.ru>*
[1,2] *М.М. Чупилко, ORCID: 0000-0002-8772-5631 <chupilko@ispras.ru>*
[1,2] *М.С. Лебедев, ORCID: 0000-0002-0207-7672 <lebedev@ispras.ru>*
[1,2] *С.А. Смолов, ORCID: 0000-0003-0173-3081 <smolov@ispras.ru>*
[2,6] *Г. Гайдаджиев, ORCID: 0000-0002-3678-7007 <g.gaydadjiev@rug.nl>*

[1] *Институт системного программирования имени В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Российский экономический университет имени Г.В. Плеханова, 117997, Россия, г. Москва, Стремянный пер., д. 36*
[3] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[4] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9*
[5] *НИУ "Высшая школа экономики", 101000, Россия, г. Москва, ул. Мясницкая, д. 20*
[6] *Гронингенский университет Нидерланды, г. Гронинген, 9700 АБ*

Аннотация. Предметно-ориентированные системы, использующие ускорители на базе программируемых логических интегральных схем (ПЛИС), все чаще проектируются при помощи инструментов высокоуровневого синтеза и конструирования аппаратуры. В настоящее время разработчикам доступно множество таких инструментов, как открытых, так и коммерческих. В данной работе проведено экспериментальное сравнение нескольких существующих решений (языков и инструментов), а именно: Verilog (базовое решение), Chisel, Bluespec SystemVerilog (Bluespec Compiler), DSLX (XLS), MaxJ (MaxCompiler) и C (Bambu и Vivado HLS). Сравнение и анализ производились на примере алгоритма обратного дискретного косинусного преобразования матрицы 8×8 (ОДКП), широко используемого, в том числе, в JPEG- и MPEG-декодерах. В качестве метрик сравнения реализаций алгоритма ОДКП использовались: 1) степень автоматизации (насколько меньше исходного кода требуется для описания алгоритма по сравнению с Verilog); 2) контролируемость (возможность достижения заданных характеристик аппаратуры, а именно отношения производительности к занимаемой площади); 3) гибкость (легкость внесения изменений в реализацию с целью достижения определенных характеристик). Для оценки характеристик синтезированных схем в реальном окружении были разработаны AXI-Stream-адаптеры. В исследовании показано, как отдельные оптимизации (настройки инструментов и модификации исходного кода) влияют на производительность системы и занимаемую площадь. Продемонстрировано, насколько важно уметь управлять балансом между пропускной способностью интерфейса и производительностью вычислительного ядра; даны рекомендации по разработке новых инструментов проектирования систем, использующих ускорители на основе ПЛИС.

Ключевые слова: автоматизация проектирования микроэлектроники; предметно-ориентированные вычисления; конструирование аппаратуры; высокоуровневый синтез; программируемые логические интегральные схемы; обратное дискретное косинусное преобразование.

Для цитирования: Камкин А.С., Чупилко М.М., Лебедев М.С., Смолов С.А., Гайдаджиев Г. Сравнение инструментов высокоуровневого синтеза и конструирования цифровой аппаратуры. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 7-22. DOI: 10.15514/ISPRAS-2022-34(5)-1

Comparison of High-Level Synthesis and Hardware Construction Tools

[1,2,3,4,5] *A.S. Kamkin, ORCID: 0000-0001-6374-8575 <kamkin@ispras.ru>*
[1,2] *M.M. Chupilko, ORCID: 0000-0002-8772-5631 <chupilko@ispras.ru>*
[1,2] *M.S. Lebedev, ORCID: 0000-0002-0207-7672 <lebedev@ispras.ru>*
[1,2] *S.A. Smolov, ORCID: 0000-0003-0173-3081 <smolov@ispras.ru>*
[2,6] *G. Gaydadjiev, ORCID: 0000-0002-3678-7007 <g.gaydadjiev@rug.nl>*

[1] *Ivannikov Institute for System Programming of the RAS,*
25 Alexander Solzhenitsyn st., Moscow, 109004, Russia
[2] *Plekhanov Russian University of Economics,*
36 Stremyanny lane, Moscow, 117997, Russia
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[4] *Moscow Institute of Physics and Technology*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
[5] *National Research University Higher School of Economics*
11 Myasnitskaya Ulitsa, Moscow, 101000, Russia
[6] *University of Groningen,*
9700 AB, Groningen, The Netherlands

Abstract. Application-specific systems with FPGA accelerators are often designed using high-level synthesis or hardware construction tools. Nowadays, there are many frameworks available, both open-source and commercial. In this work, we attempt to fairly compare several existing solutions (languages and tools), including Verilog (our baseline), Chisel, Bluespec SystemVerilog (Bluespec Compiler), DSLX (XLS), MaxJ (MaxCompiler), and C (Bambu and Vivado HLS). Our analysis has been carried out using a representative example of 8×8 inverse discrete cosine transform (IDCT), a widely used algorithm engaged in, among others, JPEG and MPEG decoders. The metrics under consideration include: (a) the degree of automation (how much less code is required compared to Verilog), (b) the controllability (possibility to achieve given design characteristics, namely a given ratio of the performance and area), and (c) the flexibility (ease of design modification to achieve certain characteristics). Rather than focusing on computational kernels only, we have developed AXI-Stream wrappers for the synthesized implementations, which allows adequately evaluating characteristics of the designs when they are used as parts of real computer systems. Our study shows clear examples of what impact specific optimizations (tool settings and source code modifications) have on the overall system performance and area. It emphasizes how important is to be able to control the balance between the communication interface utilization and the computational kernel performance and delivers clear guidelines for the next generation tools for designing FPGA accelerator based systems.

Keywords: electronic design automation; application-specific computing; hardware construction; high-level synthesis; field-programmable gate array; inverse discrete cosine transform

For citation: Kamkin A.S., Chupilko M.M., Lebedev M.S., Smolov S.A., Gaydadjiev G. Comparison of High-Level Synthesis and Hardware Construction Tools. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 7-22 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-1

1. Введение

Одним из начальных этапов проектирования цифровой аппаратуры является создание модели целевого устройства на уровне регистровых передач (register transfer level, RTL), для чего используются языки описания аппаратуры (hardware description language, HDL), такие как Verilog или VHDL [1]. Эти языки появились в 1980-90-е годы и произвели революцию в

области разработки микроэлектроники. Сейчас (несмотря на некоторое обновление в 2000-х годах) они выглядят устаревшими; например, в них нет высокоуровневых примитивов проектирования, полезных при реализации аппаратных вычислителей в некоторых предметных областях, в том числе в цифровой обработке сигналов (digital signal processing, DSP).

В настоящее время предложены два подхода, нацеленных на повышение продуктивности проектирования микроэлектронных устройств: *высокоуровневый синтез* (high-level synthesis, HLS) и *конструирование аппаратуры* (hardware construction, HC) [2, 3]. В первом случае RTL-модель автоматизированным образом синтезируется из высокоуровневого описания алгоритма. Во втором случае используется более детализированное описание, где микроархитектура устройства задается явно. В обоих случаях разработчик может проводить исследование проектных альтернатив (design space exploration, DSE) и оптимизировать модель в соответствии с заданными ограничениями на производительность, энергопотребление и занимаемую площадь.

Целями данной работы являются:

1) исследование возможностей инструментов HLS/HC для ПЛИС;
2) испытание инструментов на выбранном тестовом примере;
3) оценка эффективности полученных RTL-моделей.

В качестве тестового примера был использован алгоритм обратного дискретного косинусного преобразования (ОДКП) матрицы 8×8 [4]. Пример ОДКП был выбран по следующим соображениям: 1) это известный алгоритм, используемый в декодерах JPEG и MPEG [5, 6]; 2) это «типичная» функция одного вычислительного ядра; 3) существуют готовые реализации: на C [6], DSLX [7] и Bluespec SystemVerilog (BSV) [8]. Алгоритм ОДКП был реализован на нескольких языках, являющихся входными для инструментов HLS/HC (разработанные реализации размещены в открытом репозитории [9]). Полученные реализации (синтезированные схемы) сравнивались по ряду метрик, включая:

1) *степень автоматизации* (насколько меньше исходного кода требуется по сравнению с Verilog);
2) *управляемость* (возможность достижения заданных характеристик аппаратуры);
3) *гибкость* (легкость внесения изменений в модель и/или нахождения параметров инструментов с целью достижения заданных характеристик).

Статья организована следующим образом. В разд. 2 представлена классификация методов HLS/HC и обзор существующих HLS/HC-решений. В разд. 3 описывается методология оценки инструментов (метрики сравнения, тестовый пример и процедура оценки). В разд. 4 представлены результаты экспериментального анализа инструментов. В разд. 5 подведены итоги и описаны направления дальнейших исследований.

2. Обзор работ

Основным подходом к проектированию цифровых СБИС в настоящее время является разработка RTL-моделей на языках Verilog и VHDL (с их последующим логическим и физическим синтезом). В последнее время появляются новые языки и инструменты проектирования. Этому способствует очевидное желание снизить затраты на разработку и верификацию аппаратуры. В зависимости от способа решения этой задачи можно выделить следующие группы инструментов (в скобках приведены примеры):

1) специализированные генераторы, предназначенные для построения эффективных аппаратных реализаций определенных алгоритмов (FloPoCo [10] для арифметических ядер);

2) инструменты конструирования аппаратуры, построенные на основе языков программирования общего назначения, таких как Scala, Python и Java (Chisel [11], MyHDL [12], JHDL [13]);
3) инструменты конструирования аппаратуры на основе специализированных языков (немного более высокого уровня, чем традиционные HDL) (Bluespec Compiler [14], Esterel Studio [15], DIL Compiler [16]);
4) инструменты высокоуровневого синтеза на основе императивных языков программирования, таких как C/C++ и Fortran (Bambu [17], LegUp [18], Vivado HLS [19]);
5) инструменты высокоуровневого синтеза на основе языков параллельного программирования, таких как CUDA C, OpenCL и DPC++ (Vitis HLS [20], Intel FPGA SDK for OpenCL [21], OneAPI [22]);
6) инструменты высокоуровневого синтеза на основе языков программирования потоков данных, основанных на императивной и функциональной парадигмах (MaxCompiler [23], XLS [7]);
7) инструменты высокоуровневого синтеза на основе предметно-ориентированных языков и библиотек, таких как MATLAB, TensorFlow и OpenVX (HDL Coder [24], Vitis AI [25], HiFlipVX [26]).

В последние несколько лет было опубликовано несколько обзоров инструментов HLS/HC [27, 28, 29]. В работе [27] рассмотрены следующие инструменты: AccelDSP, Agility, AutoPilot, Bluespec Compiler (BSC), Catapult C, Compaan, C-to-Silicon, CyberWorkBench, DK Design Suite, Impulse CoDeveloper, ROCCC и Synphony C. Сравнение инструментов проводилось по следующим характеристикам:

1) язык описания и уровень абстракции;
2) легкость разработки и процесса обучения;
3) поддержка чисел с плавающей и фиксированной точкой;
4) легкость исследования проектных альтернатив (опции оптимизации, модификация исходного кода и т.д.);
5) возможности для верификации (наличие/отсутствие встроенных генераторов тестовых окружений);
6) использованные для синтезированных схем ресурсы ПЛИС.

В качестве примера использовался фильтр Собеля 3×3. На данный момент этот обзор несколько устарел (появились новые инструменты). Кроме того, обзор посвящен в основном оценке продуктивности разработки; в нем отсутствует информация, актуальная для нашего исследования – насколько хорошо инструменты могут оптимизировать по сравнению с RTL-моделями, разработанными вручную.

В работе [28] авторы концентрируются на использовании инструментов HLS/HC для разработки гетерогенных компьютерных систем, использующих ускорители на основе ПЛИС. Существующие инструменты были разделены на 4 категории в зависимости от языка описания:

1) C/C++ и их подмножества: Trident, GAUT, Streams-C, Impulse-C, FpgaC, NAPA C, Nimble, CHiMPS, CASH, LegUp, ROCCC, C2H, ASC, Catapult C, AutoPilot, DEFACTO, Carte, DIME-C, Bash-C, Mitrion-C, SpecC, SPC и SPARK;
2) не C/C++-подобные языки: MATLAB (MATCH), Python (MyHDL), Java (JHDL, Sea Cucumber), C# (Kiwi), Pebble, Esterel и BSV;
3) языки визуального моделирования: LabVIEW, Simulink и Altium Designer;
4) языки, ориентированные на графические процессоры: CUDA (FCUDA) и OpenCL (SOpenCL).

В обзоре отсутствует сравнение инструментов HLS/HC друг с другом. Многие упомянутые в нем средства либо уже не поддерживаются, либо не доступны для скачивания.

В работе [29] авторы провели подробный анализ актуальных инструментов HLS/HC и оценили некоторые из них (компиляторы из C в Verilog/VHDL). В обзоре рассмотрены инструменты eXCite, Cynthesizer, CHC, MaxCompiler, Bambu, DWARV, Vivado HLS, gcc2verilog, HercuLeS, Garp, PipeRench, SA-C, CtoVerilog, а также средства, перечисленные в статьях [27, 28]. Сравнение проводилось по следующим характеристикам:

1) тип лицензии;
2) входной и выходной языки;
3) область применения;
4) автоматизация тестирования;
5) поддержка чисел с фиксированной и плавающей точкой.

Отличительная особенность работы – описание базовых оптимизаций HLS/HC. В статье представлены результаты экспериментального сравнения трех академических инструментов (Bambu, DWARV и LegUp) и одного коммерческого (не назван). Авторы провели два набора экспериментов на выбранных тестовых примерах: сначала инструменты синтеза запускались с настройками по умолчанию; затем примеры модифицировались с целью максимизации производительности и инструменты синтеза запускались на измененных примерах с применением опций оптимизации. В обоих случаях для синтезированных RTL-моделей оценивались тактовая частота, общая задержка и количество использованных LUT, DSP-блоков и BRAM. Большинство тестовых примеров было взято из набора CHStone [30].

В заключение обзора отметим, что существующие обзоры либо устарели, либо содержат только общие сведения о возможностях инструментов HLS/HC, либо описывают эксперименты для небольшого набора инструментов. Данная работа призвана устранить эти недостатки.

3. Методология сравнения

В данном разделе описывается методология оценки инструментов HLS/HC.

3.1 Метрики

Для сравнения используются следующие первичные показатели:

- *размер исходного кода* (L) – количество строк кода (lines of code, LOC), включая настройки соответствующих инструментов HLS/HC (директивы, параметры, опции и т.д.);
- *производительность*, или *пропускная способность* (P), – количество операций в секунду (operations per second, OPS);
- *площадь*, или *объем использованных ресурсов* (A), – количество задействованных LUT и триггеров (flip-flop, FF) на ПЛИС.

Количество строк кода можно считать объективной метрикой, хотя и не идеальной. По всей видимости, она может быть адаптирована к языкам визуального программирования (т.к. инструменты HLS/HC должны генерировать код на каком-то этапе), хотя они не рассматриваются в данной работе.

Отношение $Q = \frac{P}{A}$ будем называть *качеством* модели, или *эффективностью*. Пусть Ф – целевая функция (далее будем считать, что $\Phi \equiv Q$).

Пусть $\mathcal{A}$ – алгоритм (тестовый пример), $\mathcal{L}$ – язык описания аппаратуры и $\mathcal{T}$ – инструмент HLS/HC на основе $\mathcal{L}$.

1) *Степень автоматизации*: α показывает, насколько легко (в терминах количества строк кода) описать $\mathcal{A}$ с помощью $\mathcal{L}$ и настроить $\mathcal{T}$ в сравнении с ручной разработкой на Verilog (L_V):

$$\alpha = \frac{(L_V - L)}{L_V} \times 100\%. \quad (1)$$

2) *Управляемость*: набор средств, позволяющих разработчику аппаратуры влиять на результаты синтеза (в терминах Ф) без изменения функциональности. Включает в себя настройки инструмента, аннотации кода и сам исходный код. Метрика управляемости C_Φ определяется по следующей формуле:

$$C_\Phi = \frac{\Phi^*}{\Phi_V^*} \times 100\%, \quad (2)$$

где Φ^* – максимальное значение Ф, полученное инструментом $\mathcal{T}$, а Φ^*_V – «абсолютный» максимум (ожидаемый при использовании Verilog-модели, разработанной вручную).

3) *Гибкость*: F_Φ показывает, насколько легко улучшить значение Ф путем настройки $\mathcal{T}$ и изменения исходного кода:

$$F_\Phi = \frac{\Phi^* - \Phi_0}{\Delta L}, \quad (3)$$

где значения Φ^* и Φ_0 соответствуют «оптимальному» и «начальному» вариантам модели, а $\Delta L = \Delta L^+ + \Delta L^-$ – это количество измененных строк кода (добавленных или удаленных), включая аннотации и опции инструмента.

3.2 Тестовый пример

В качестве тестового примера используется ОДКП матрицы 8×8. Преобразование определяется следующей формулой:

$$f_{ij} = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C_u C_v F_{uv} \cos\left(\frac{\pi}{8}\left(i+\frac{1}{2}\right)u\right)\cos\left(\frac{\pi}{8}\left(j+\frac{1}{2}\right)v\right), \quad (4)$$

где $F \in \mathbb{R}^{8\times8}$ – входное изображение в частотной области, $f \in \mathbb{R}^{8\times8}$ – выходное изображение в пространственной области и

$$C_k = \begin{cases} 1/\sqrt{2}, k = 0, \\ 1, \quad k > 0. \end{cases}$$

Формулу (4) можно представить в виде 16 одномерных преобразований: 8 по строкам и 8 по столбцам матрицы. Каждое из них определяется формулой:

$$g_i = \frac{1}{2}\sum_{u=0}^{7} C_u G_u \cos\left(\frac{\pi}{8}\left(i+\frac{1}{2}\right)u\right), \; i = \overline{0,7}. \quad (5)$$

Алгоритм ОДКП 8×8 представлен ниже:

for $i = \overline{0,7}$ **do** $temp_{i*} \leftarrow \text{IDCT}^{row}(F_{i*})$ **end** // по строкам
for $i = \overline{0,7}$ **do** $f_{ij} \leftarrow \text{IDCT}^{col}(temp_{*j})$ **end** // по столбцам

Алгоритм 1. ОДКП 8×8
Algorithm 1. IDCT 8×8

Здесь x_{k*} (x_{*k}) означает k-ю строку (столбец) матрицы x. На вход алгоритму подается матрица 12-битных чисел; на выходе формируется матрица 9-битных чисел:

$$F_{uv} \in [-2{,}048, 2{,}047], \qquad f_{ij} \in [-256, 255].$$

Рассматриваемые реализации следуют представленной выше схеме, но вместо применения формулы (5) для IDCT^{row} и IDCT^{col} они используют алгоритм Чена-Вана с использованием «бабочки» (11 умножений, 29 сложений и несколько сдвигов) [31, 32]. Все реализации совместимы со стандартом IEEE 1180-1990 [4] и разрабатывались на основе реализации ОДКП из набора тестов на соответствие ISO/IEC 13818-4:2004 [6, 33].

Заметим также, что IDCT^{row} и IDCT^{col} используют разные коэффициенты: $C_k^{row} = C_k \times 256\sqrt{2}$ и $C_k^{col} = \frac{C_k}{256\sqrt{2}}$. Все реализации, разработанные в рамках данного исследования, размещены в открытом репозитории [9].

3.3 Процедура сравнения

Для всех рассмотренных в данном исследовании нотаций (Verilog, Chisel, BSV, DSLX, MaxJ, C) «начальная» модель была разработана в соответствии с алгоритмом [6]. Для получившегося функционального блока вычислялось количество строк исходного кода (без учета комментариев и пустых строк) – L^{FU}. «Оптимальная» модель (для целевой функции Ф) разрабатывалась путем применения опций и/или прагм инструментов и модификации исходного кода. Для оптимизации моделей применялась конвейеризация вычислений.

Для того, чтобы получить AXI-Stream-совместимые [34] модели (что распространено для библиотек IP-блоков [35]), были разработаны адаптеры интерфейса. Входные и выходные матрицы передаются построчно через вход `TDATA` (с помощью протокола с подтверждением установления связи через `TVALID/TREADY`). Адаптеры создавались либо автоматически инструментом HLS/HC, либо вручную на языке разработки или Verilog. Для второго случая подсчитывалось количество строк кода адаптера – L^{AXI}. Единственное отступление от данной процедуры было сделано для языка MaxJ и инструмента MaxCompiler. Последний работает только на системном уровне и генерирует PCIe-адаптер. В этом случае рассматривалось вычислительное ядро без адаптера интерфейса ($L^{AXI} = 0$).

Если инструмент требовал использования дополнительных настроек, то учитывались размер конфигурационных файлов и количество параметров – L^{Conf}. Для экспериментов с начальными версиями описаний применялись настройки по умолчанию ($L^{Conf} = 0$).

Трудозатраты на создание функционального блока вычислялись как $L = L^{FU} + L^{AXI} + L^{Conf}$. Степень автоматизации (α) вычислялась по формуле (1).

Для синтеза на ПЛИС использовался коммерческий инструмент Vivado Design Suite 2017.4 [19] с настройками по умолчанию. Для оценки производительности измерялось минимальное значение тактового сигнала (T_{clk}), для которого синтез завершался успешно, и вычислялись максимальная частота и пропускная способность:

$$v_{max} = \frac{1}{(T_{clk} - T_{wns})}, \quad P = \frac{v_{max}}{T_P},$$

где T_{wns} – *максимальное время запаздывания* (worst negative slack, WNS), вычисленное Vivado, и T_P – *периодичность*, т.е. минимальное количество тактов между запусками двух последовательных операций.

В экспериментах также измерялась *задержка* (T_L), т.е. количество тактов исполнения одной операции (включая передачу входных/выходных данных).

Для оценки площади, занимаемой синтезированной схемой на ПЛИС, брались следующие показатели:

- N_{LUT} – количество LUT;
- N_{FF} – количество триггеров;
- N_{DSP} – количество DSP-блоков;
- N_{BRAM} – количество блоков памяти BRAM;
- N_{IO} – количество входных/выходных пинов.

Так как при физическом синтезе различные модели по-разному отображаются на ресурсы ПЛИС (особенно на DSP-блоки), использовался нормализованный показатель:

$$A = N_{LUT}^* + N_{FF}^*,$$

где N^*_{LUT} и N^*_{FF} – количество задействованных LUT и триггеров соответственно при запрете использования DSP-блоков инструментом синтеза (опция `maxdsp=0` в Vivado).

Для полученных значений P и A вычислялось значение целевой функции Ф. Для оценки возможностей инструментов HLS/HC модели оптимизировались с целью максимизации Ф, после чего определялись показатели $C_Ф$ и $F_Ф$ по формулам (2) и (3).

Табл. 1. Исследуемые языки и инструменты
Table 1. Languages and tools under evaluation

Язык	Парадигма	Инструмент	Тип	Лицензия
Verilog	Классический RTL	Vivado	LS/PR	Коммерческая
Chisel	Функциональный/RTL	Chisel	HC	Открытая
BSV	Основанный на правилах/RTL	BSC	HC	Открытая
DSLX	Функциональный	XLS	HLS	Открытая
MaxJ	Потоковый	MaxCompiler	HLS	Коммерческая
C	Императивный	Bambu	HLS	Открытая
		Vivado HLS	HLS	Коммерческая

4. Экспериментальный анализ

Рассмотренные языки и инструменты представлены в табл. 1 (где LS/PR – logic synthesis/place & route, т.е. *логический синтез/размещение и трассировка*). Во всех экспериментах для синтеза схем, оценки их характеристик и генерации двоичных образов для ПЛИС был использован инструмент Vivado Design Suite. Синтез проводился для ПЛИС Xilinx Virtex Ultrascale+ (XCVU9P-FLGB2104-2-E), имеющей следующие характеристики: N_{LUT} = 1182240, N_{LUTRAM} = 591840, N_{FF} = 2364480, N_{DSP} = 6840, N_{BRAM} = 2160, N_{IO} = 702.

Результаты анализа сгруппированы по входным языкам (см. ниже). Количественные данные представлены на рис. 1 и в табл. 2.

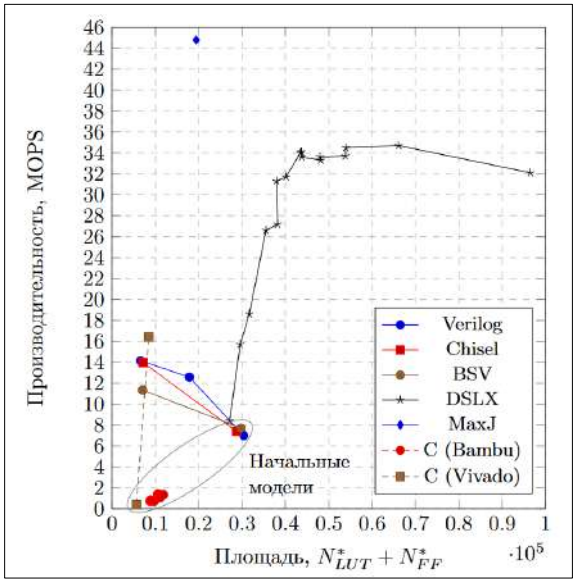


Рис. 1. Исследование проектных альтернатив для ОДКП
Fig. 1. Design space exploration for IDCT

4.1 Verilog

Verilog – язык описания аппаратуры, созданный в середине 1980-х годов и предназначенный для проектирования цифровых СБИС на уровне регистровых передач [36]. Модель представляет собой иерархию модулей; каждый из которых имеет входные и выходные порты и содержит объявления переменных, параллельные процессы (блоки `always` и непрерывные присваивания) и экземпляры других модулей. Для переменных требуется явно указывать их разрядность. Блоки описываются в С-подобном синтаксисе. Язык поддерживает параметризацию модулей и конструкцию `generate` для порождения экземпляров модулей.

Начальная модель. Сначала была разработана простая комбинационная схема с построчным накоплением входных данных через AXI-Stream-адаптер. Схема обрабатывает матрицу целиком и содержит восемь экземпляров модуля $IDCT^{row}$ и восемь экземпляров модуля $IDCT^{col}$, что приводит к большой занимаемой площади (30 396) и низкой частоте (55,88 МГц). Самым узким местом в этой реализации является последовательный адаптер (теоретически, пропускная способность схемы в 8 раз выше).

Оптимизации. Были разработаны две оптимизированные модели. Первая состоит из одного экземпляра $IDCT^{row}$ и восьми $IDCT^{col}$ (поскольку за один такт работы схемы на вход поступает только одна строка данных, то достаточно одного экземпляра $IDCT^{row}$). Пропускная способность выросла в 1,8 раза (за счет повышения частоты), а площадь уменьшилась в 1,7 раза. Соответственно, качество выросло более чем в 3 раза.

Вторая реализация состоит из одного экземпляра $IDCT^{row}$ и одного $IDCT^{col}$. Задержка схемы выросла с 17 до 24 тактов, в то время как пропускная способность увеличилась в 2 раза (по сравнению с начальной моделью), площадь уменьшилась в 4,6 раза, а качество выросло в 9,4 раза.

4.2 Chisel

Chisel (Constructing Hardware in a Scala Embedded Language) [11] – язык описания аппаратуры на уровне регистровых передач, основанный на Scala. В добавление к обычным конструкциям языка описания аппаратуры, он предоставляет возможности объектно-ориентированного и функционального программирования (полиморфизм, абстрактные типы данных, рекурсия), что позволяет разрабатывать на нем гибкие генераторы компонентов СБИС [3]. Язык поддерживает вывод типов: разрядность портов и регистров задается вручную, а размеры остальных переменных могут быть вычислены автоматически. Следует заметить, что Chisel работает в связке с промежуточным представлением FIRRTL (Flex Intermediate Representation for RTL) [37].

Начальная модель. Как и в случае с Verilog, сначала была реализована простая комбинационная схема. Данная реализация имеет несколько большую производительность по сравнению с Verilog (105,7%) и при этом меньшую площадь (94,6%). Это можно объяснить тем, что в Verilog-описании используется 32-битная арифметика (как в [6]), в то время как Chisel автоматически (и точнее) определяет разрядность переменных.

Оптимизации. Была разработана конвейерная реализация с одним модулем $IDCT^{row}$ и одним модулем $IDCT^{col}$. Она оказалась немного хуже аналогичной Verilog-реализации (производительность – 98,7%, площадь – 109,5%). Следует отметить, что модель на Chisel была разработана в 2-3 раза быстрее.

4.3 Bluespec SystemVerilog (BSC)

Bluespec SystemVerilog (BSV) – основанный на правилах язык описания аппаратуры, разработанный в начале 2000-х [14]. BSV-описания являются иерархическими, но модули описываются не так, как на Chisel или Verilog. В каждом модуле определены элементы состояния и правила, меняющие это состояние. Правило состоит из условия срабатывания и

атомарного действия, описывающего изменение состояния. BSV находится на более высоком уровне абстракции по сравнению с Verilog и Chisel (в терминах описания времени): модель программирования (операционная семантика) подразумевает выполнение одного правила за такт, однако компилятор может оптимизировать модель и запланировать выполнение нескольких правил на один такт.

Были разработаны две модели на BSV. По ним были синтезированы 26 реализаций путем варьирования опций инструмента BSC и применения атрибутов в коде. Оказалось, что настройки оказывают небольшое влияние (менее 1%) на производительность и площадь, поэтому данные ниже соответствуют настройкам инструмента по умолчанию.

Начальная модель. Начальная BSV-реализация была разработана на основе C-реализации [6]. Полученная модель оказалась несколько лучше начальных реализаций на Verilog и Chisel (производительность – 110,3%, площадь – 97,2% по сравнению с Verilog).

Оптимизации. Оптимизированная версия также основана на конвейеризации. Ее качество оказалось немного ниже: производительность – 80,2%, а площадь – 107,1% относительно Verilog. Главная причина меньшей производительности – в том, что показатель периодичности на один такт больше (9 вместо 8). Теоретически данная особенность может быть устранена, что позволит получить характеристики, сравнимые с характеристиками Chisel-реализации.

4.4 DSLX (XLS)

DSLX – потоковый функциональный язык для описания вычислительных ядер [38]. Он похож на язык Rust, но учитывает аппаратные особенности (объекты фиксированного размера, полностью анализируемые графы вызовов). Поддерживаемые типы данных: битовые векторы, кортежи, структуры и массивы. Можно отметить другие детали: параметрические структуры и функции, вывод типов, конструкции `for` (циклы с фиксированным числом итераций). Будучи высокоуровневым языком, DSLX игнорирует временны́е свойства, позволяя компилятору планировать вычисления (разбивать граф вычислений по стадиям конвейера).

Для экспериментов был адаптирован существующий пример ОДКП [7] (были изменены разрядности элементов входной и выходной матриц). Были синтезированы 19 реализаций путем применения следующих опций: 1) тип схемы (комбинационная или конвейерная) и 2) количество стадий конвейера.

Начальная модель: Сначала была создана комбинационная схема с AXI-Stream-адаптером, разработанным вручную. Эта реализация заметно лучше, чем все начальные реализации, представленные ранее: производительность – 120,3%, а площадь – 89,2% относительно Verilog.

Оптимизации: Эксперименты показали, что максимальное качество достигается, когда количество стадий конвейера равно 8 (по неизвестным причинам исполнение операции при этом занимает 3 такта): производительность – 221,2%, а площадь – 578,1% по сравнению с конвейерной Verilog-реализацией. Показатель качества в 38,3% выглядит не очень хорошо; вероятно, проблема в последовательном адаптере интерфейса (пропускная способность могла бы быть в 8 раз выше).

4.5 MaxJ (MaxCompiler)

MaxJ – потоковый императивный язык программирования для разработки высокопроизводительных систем [39]. Системы, проектируемые на этом языке, состоят из трех частей: 1) вычислительные ядра, 2) менеджер (соединяющий ядра друг с другом, ЦПУ и памятью) и 3) управляющее программное обеспечение. Ядра и менеджер описываются на MaxJ. Язык основан на Java и предназначен для описания графов потоков данных (dataflow graphs), в которых могут быть узлы следующих типов: значения (константы и параметры

времени исполнения), вычислительные узлы (арифметические и логические операции), смещения (доступ к “прошлым” и “будущим” элементам потоков данных), мультиплексоры (узлы принятия решений), счетчики (организация циклов) и входы/выходы.

В отличие от других рассмотренных инструментов, MaxCompiler позволяет разработать систему целиком, а не только аппаратную часть на ПЛИС; предоставляет программный интерфейс и библиотеку времени исполнения для передачи данных на вычислительные ядра и управления вычислениями. Программная и аппаратная части соединены через шину PCIe. Таким образом, не совсем корректно сравнивать MaxJ-реализации с другими реализациями.

Начальная модель. Сначала была разработана модель, которая на каждом такте принимает и выдает матрицу 8×8 целиком. Синтезированная реализация содержит 47 конвейерных стадий и может работать на частоте 403,13 МГц, самой высокой среди всех реализаций. Здесь узким местом является шина PCIe. Пропускная способность оценивается как пропускная способность шины PCIe 3.0 x 16 (около 16 ГБ/с), поделенная на размер входных данных (1024 бита). Несмотря на то, что реализация требует много ресурсов, ее качество (963%) существенно выше, чем у начальной Verilog-реализации.

Оптимизации. Для снижения занимаемой площади было разработано другое вычислительное ядро, которое за такт получает на вход одну строку матрицы и сохраняет промежуточные результаты в памяти ПЛИС. В отличие от начального варианта, в этом случае производительность ограничена частотой. Занимаемая площадь сократилась в 2,8 раза, пропускная способность – в 2,7 раза, а качество повысилось на 4%.

4.6 C (Bambu и Vivado HLS)

C [40] – императивный язык программирования, разработанный в начале 1970-х годов. Язык широко применяется в различных областях, включая системное программирование, вычислительную математику и цифровую обработку сигналов. Базовыми элементами программ на языке C являются объявления данных и функции. Язык поддерживает целые числа и числа с плавающей точкой, а также указатели, массивы, структуры и объединения; есть условные операторы и циклы. Так как C предполагает один поток исполнения, по C-программам сложно синтезировать высокопроизводительные устройства.

Были проведены эксперименты с двумя компиляторами из C в Verilog: открытым Bambu и коммерческим Vivado HLS. В обоих случаях C-код [6] был модифицирован: округление в $IDCT^{col}$ было реализовано в виде функции (`iclip`), а не массива фиксированных значений [9]. Bambu, в отличие от Vivado HLS, не умеет генерировать AXI-Stream-адаптеры, поэтому адаптер был разработан вручную на Verilog.

4.6.1 Bambu

Инструмент Bambu предоставляет широкий набор опций, покрывающий все этапы высокоуровневого синтеза: планирование операций, выделение ресурсов и т.д. В данном исследовании применялся набор настроек `experimental-setup`, предназначенный для синтеза схем, оптимальных по разным критериям (производительность, площадь и баланс между ними). Было установлено, что большинство опций не оказывает ощутимого влияния на качество синтезируемых схем. Также был обнаружен странный эффект – хранение данных в BRAM (поведение по умолчанию) приводит к некорректному функционированию модели (при отключении этой опции модель работает корректно). Всего были исследованы 42 конфигурации.

Начальная модель. Сначала была использована следующая конфигурация: `channels-type=MEM_ACC_11` (один канал на чтение и один канал на запись в память) и `memory-allocation-policy=LSS` (локальные/статические переменные и строки хранятся в BRAM). Несмотря на относительно высокую частоту (471,4% относительно начальной

Verilog-реализации), схема, будучи последовательной, имеет низкую производительность (11,7%) при небольшой площади (29,2%).

Оптимизации. Наилучшее качество было достигнуто на конфигурации `BAMBU-PERFORMANCE-MP` (два канала на чтение и два канала на запись в память) в сочетании с опциями `speculative-sdc-scheduling` и `memory-allocation-policy=LSS`. Полученные характеристики близки к показателям начальной модели и существенно хуже характеристик оптимизированных реализаций, построенных по моделям на других языках: производительность – 9,8%, площадь – 160,1% по сравнению с Verilog.

4.6.2 Vivado HLS

В инструменте Vivado HLS процесс синтеза управляется директивами препроцессора (прагмами). Существуют директивы конвейеризации, встраивания функций, развертки циклов, оптимизации массивов, упаковки структур, синтеза интерфейса и т.д. Отметим, что инструмент может автоматически генерировать требуемый интерфейс. Для создания AXI-Stream-адаптера была добавлена отдельная функция и прагма `#pragma HLS INTERFACE axis port=⟨name⟩`.

Начальная модель. Реализация, синтезированная с настройками по умолчанию, имеет не слишком высокие характеристики: пропускная способность почти в 18 раз меньше, чем у начальной Verilog-реализации. Основная проблема состоит в том, что инструмент не подставляет код функций $IDCT^{row}$ и $IDCT^{col}$ в места их вызова и генерирует лишние AXI-Stream-интерфейсы между ними.

Оптимизации. Для преодоления описанной выше проблемы исходный код был модифицирован: массив `short buf[8]` в $IDCT^{row}$ и $IDCT^{col}$ был заменен на явно заданную последовательность элементов `short buf0, … short buf7`. Также была добавлена прагма `HLS PIPELINE`. В результате, качество синтезированной реализации вплотную приблизилось к оптимизированной Verilog-реализации (89,7%): производительность – 116,1%, площадь – 129,5%.

Табл. 2. Результаты оценки инструментов HLS/HC
Table 2. HLS/HC tools evaluation results

Язык	**Verilog**		**Chisel**		**BSV**		**DSLX**	
Инструмент	**Vivado**		**Chisel**		**BSC**		**XLS**	
Конфигурация	*Нач.*	*Опт.*	*Нач.*	*Опт.*	*Нач.*	*Опт.*	*Нач.*	*Опт.*
LOC, с опциями	247	316	195	222	238	199	242	243
Модификации, Δ*L*	258		131		434		**3**	
Автоматизация, *α*	0%	0%	21,1%	29,8%	3,6%	37,0%	2,0%	23,1%
Качество, *Q = P/A*	230	2155	257	1942	259	1614	310	825
Контролируемость, C_Q	100%		90.1%		74.8%		38.3%	
Гибкость, F_Q	7.5		12.9		3.1		**171.7**	
Частота, МГц	55,88	113,21	59,15	111,77	100,25	102,18	67,30	250,50
Производительность, MOPS	6,99	14,15	7,39	13,97	7,71	11,35	8,41	31,31
Задержка, тактов	**17**	24	**17**	24	21	26	**17**	19
Периодичность, тактов	8	8	8	8	13	9	8	8
Площадь, N_{LUT}^*+ N_{FF}^*	30396	6567	28778	7194	29549	7036	27127	37965
N_{LUT}^* (maxdsp=0)	29059	**3909**	27441	4530	27565	4781	25805	26960
N_{FF}^* (maxdsp=0)	1337	2658	1337	2664	2184	2255	**1322**	11005
N_{LUT}	13850	**2106**	9283	2205	26560	4643	25805	26010
N_{FF}	1337	2658	1337	2661	2184	2255	**1322**	10717
N_{DSP}	160	20	184	23	40	4	0	16
N_{IO}	172	170	172	172	174	172	170	170

Язык	**MaxJ**		**C**			
Инструмент	**MaxCompiler**		**Bambu**		**Vivado HLS**	
Конфигурация	*Нач.*	*Опт.*	*Нач.*	*Опт.*	*Нач.*	*Опт.*
LOC, с опциями	**121**	163	183	191	125	130
Модификации, ΔL	231		29		71	
Автоматизация, α	51,4%	48,4%	25,9%	39,6%	49,0%	**58,9%**
Качество, $Q = P/A$	2215	**2308**	91	132	69	1933
Контролируемость, C_Q	**100% [107,1%]**		6,1%		89,7%	
Гибкость, F_Q	0,4		1,4		26,3	
Частота, МГц	**403,13**	**403,13**	263,44	257,33	132,61	131,46
Производительность, MOPS	**123,08**	44,79	0,82	1,39	0,39	16,43
Задержка, тактов	47	60	323	185	340	26
Периодичность, тактов	1	9	323	185	340	8
Площадь, $N_{LUT}^* + N_{FF}^*$	55580	19413	8879	10514	**5633**	8501
N_{LUT}^* (maxdsp=0)	19704	5941	5443	7134	4103	4974
N_{FF}^* (maxdsp=0)	35876	13472	3436	3380	1530	3527
N_{LUT}	19704	5941	5090	6614	2334	3123
N_{FF}	35876	13472	3436	3156	1481	3346
N_{DSP}	384	48	5	9	22	19
N_{IO}	59	59	174	174	270	267

4.7 Резюме

Данные, полученные в результате экспериментов, представлены на рис. 1 и в табл. 2. Рисунок показывает результаты исследования проектных альтернатив в пространстве *Производительность*×*Площадь*. В таблице содержится информация о начальных и оптимизированных реализациях, полученных с помощью разных инструментов HLS/HC. Жирным цветом выделены наилучшие значения характеристик.

Максимальный уровень *автоматизации* достигается инструментами MaxCompiler и Vivado HLS. Первый – это специализированный инструмент; второй можно считать наилучшим решением для быстрого прототипирования. Говоря об *управляемости* (и *качестве* полученных реализаций), можно отметить инструменты Chisel и BSC; Vivado HLS тоже показывает хороший результат. Самые *гибкие* инструменты – это XLS и Vivado HLS, хотя они конфигурируются по-разному: у первого используется всего один параметр (количество стадий конвейера), у второго – прагмы в исходном коде.

Пример ОДКП, в силу своей простоты, использует немного ресурсов ПЛИС; таким образом, полученные результаты не могут быть однозначно экстраполированы на более сложные модели. Для сложных схем, требующих практически всех ресурсов ПЛИС, инструменты размещения и трассировки ведут себя по-другому, пытаясь «вписать» схему в ПЛИС с помощью различных оптимизаций. Данная работа посвящена более ранним этапам проектирования.

5. Заключение

В статье рассмотрены инструменты высокоуровневого синтеза и конструирования аппаратуры, использующие разные входные нотации: Chisel (основанный на Scala язык описания на уровне регистровых передач), BSC (язык спецификации аппаратуры на базе правил), XLS (Rust-подобный функциональный язык), MaxCompiler (основанный на Java язык описания потоков данных), Bambu и Vivado HLS (оба поддерживают язык общего назначения C). Для каждого инструмента на примере ОДКП были оценены уровень автоматизации, управляемость и гибкость.

Исследование показало, что самым сбалансированным решением (среди рассмотренных) является коммерческий инструмент Vivado HLS. Другим перспективным решением может быть использование открытых расширяемых платформ. Концепция такого инструмента может выглядеть следующим образом. На верхнем уровне находятся *предметно-ориентированные языки* и связанные с ними механизмы трансляции и оптимизации. Далее идут *языки общего назначения*, ориентированные на параллельные вычисления (например, MaxJ) и соответствующее промежуточное представление (вычислительный граф). Отдельные блоки могут быть разработаны с помощью *низкоуровневых инструментов*, как универсальных (XLS, Chisel, BSC, Verilog), так и специализированных (FloPoCo). Важной функцией является способность генерировать внешние и внутренние интерфейсы (как в Vivado HLS). Разработка такого инструмента – основная цель нашей дальнейшей работы.

Список литературы / References

[1] Botros N.M. HDL Programming Fundamentals: VHDL and Verilog. Charles River Media, 2005, 506 p.

[2] Coussy P., Gajski D.D. et al. An introduction to high-level synthesis. IEEE Design & Test of Computers, vol. 26, issue 4, 2009, pp. 8-17.

[3] Bachrach J., Vo H. et al. Chisel: constructing hardware in a Scala embedded language. In Proc. of the Design Automation Conference (DAC), 2012, pp. 1216-1225.

[4] IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform. In IEEE Std 1180-1990, 1991, pp. 1-12.

[5] Information technology – Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF) – Part 5: ISO/IEC 10918-5:2013, 2013.

[6] MPEG-2 decoder from ISO/IEC 13818-4:2004. Available at: https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_13818-4_2004_Conformance_Testing/Video/verifier/mpeg2decode_960109.tar.gz, accessed 02.11.2022.

[7] XLS. Available at: https://github.com/google/xls, accessed 02.11.2022.

[8] Bluespec Compiler. Available at: https://github.com/B-Lang-org/bsc, accessed 02.11.2022.

[9] IDCT algorithm implementations. Available at: https://github.com/ispras/hls-idct, accessed 02.11.2022.

[10] de Dinechin F., Pasca B. Designing custom arithmetic data paths with FloPoCo. IEEE Design & Test of Computers, vol. 28, issue 4, 2011, pp. 18-27.

[11] Chisel. Available at: https://github.com/chipsalliance/chisel3, accessed 02.11.2022.

[12] MyHDL. Available at: https://www.myhdl.org, accessed 02.11.2022.

[13] Bellows P., Hutchings B. JHDL-an HDL for reconfigurable systems. In Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines, 1998, pp. 175-184.

[14] Bluespec SystemVerilog Reference Guide, Bluespec, Inc., 2017, 421 p.

[15] Berry G., Gonthier G. The ESTEREL synchronous programming language: design, semantics, implementation. Science of Computer Programming, vol. 19, issue 2, 1992, pp. 87-152.

[16] Goldstein S., Budiu M. Fast compilation for pipelined reconfigurable fabrics. In Proc. of the ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays, 1999, pp. 195-205.

[17] Bambu. Available at: https://github.com/ferrandi/PandA-bambu, accessed 02.11.2022.

[18] LegUp. Available at: http://legup.eecg.utoronto.ca, accessed 02.11.2022,

[19] SmartHLS Compiler. Available at: https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/smarthls-compiler, accessed 02.11.2022.

[20] Vivado Design Suite User Guide Implementation. UG904 (v2021.1), August 30, 2021. Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_1/ug904-vivado-implementation.pdf, accessed 02.11.2022.

[21] Vitis High-Level Synthesis User Guide. UG1399 (v2021.2), December 15, 2021. Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_2/ug1399-vitis-hls.pdf, accessed 02.11.2022.

[22] Intel FPGA SDK for OpenCL. Available at: https://www.intel.com/content/www/us/en/software/programmable/sdk-for-opencl/overview.html, accessed 02.11.2022.

[23] Intel oneAPI. Available at: https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html, accessed 02.11.2022.

[24] MaxCompiler. Available at: https://www.maxeler.com/products/software/maxcompiler, accessed 02.11.2022.

[25] HDL Coder. Available at: https://www.mathworks.com/products/hdl-coder.html, accessed 02.11.2022.

[26] Vitis AI. Available at: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html, accessed 02.11.2022.

[27] Kalms L., Podlubne A., Göhringer D. HiFlipVX: an Open Source High-Level Synthesis FPGA Library for Image Processing. Lecture Notes in Computer Science, vol. 11444, 2019, pp. 149-164.

[28] Meeus W., Van Beeck K. et al. An overview of today's high-level synthesis tools. Design Automation for Embedded Systems, vol. 16, 2012, pp. 31-51.

[29] Daoud L., Zydek D., Selvaraj H. A survey of high level synthesis languages, tools, and compilers for reconfigurable high performance computing. Advances in Intelligent Systems and Computing, vol. 240, 2014, pp. 483-492.

[30] Nane R., Sima V.-M. et al. A survey and evaluation of FPGA high-level synthesis tools. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, issue 10, 2016, pp. 1591-1604.

[31] Hara Y., Tomiyama H. et al. CHStone: A benchmark program suite for practical C-based high-level synthesis. In Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS), 2008, pp. 1192–1195.

[32] Chen W.H., Smith C.H., Fralick S.C. A fast computational algorithm for the discrete cosine transform. IEEE Transactions on Communications, vol. 25, issue 9, 1977, pp. 1004-1009.

[33] Wang Z. Fast algorithm for the discrete W transform and for the discrete Fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, issue 4, 1984, pp. 803-816.

[34] Information technology – Generic coding of moving pictures and associated audio information – Part 4: Conformance testing, ISO/IEC 13818-4:2004, December 2004.

[35] AMBA 4 AXI4-Stream Protocol Specification. ARM, Cambridge, UK, ARM IHI 0051A (ID030610), March 03, 2010. Available at: https://developer.arm.com/documentation/ihi0051/a/, accessed 02.11.2022.

[36] Vivado Design Suite User Guide: Model-Based DSP. Design Using System Generator. UG897 (v2020.2), November 18, 2020. Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2020_2/ug897-vivado-sysgen-user.pdf, accessed 02.11.2022.

[37] IEEE Standard for Verilog Hardware Description Language, IEEE Std 1364-2005, April 7, 2006, 509 p.

[38] FIRRTL. Available at: https://github.com/chipsalliance/firrtl, accessed 02.11.2022.

[39] DSLX Reference. Available at: https://google.github.io/xls/dslx_reference, accessed 02.11.2022.

[40] Multiscale Dataflow Programming. Maxeler Technologies, London, UK, Version 2021.1, May 14, 2021.

[41] Information technology – Programming languages – C, ISO/IEC 9899:2018, June 2018.

Информация об авторах / Information about authors

Александр Сергеевич КАМКИН – кандидат физико-математических наук, ведущий научный сотрудник отдела технологий программирования ИСП РАН, ведущий научный сотрудник научной лаборатории «Гетерогенные компьютерные системы» РЭУ им. Г.В. Плеханова; преподает в МГУ им. М.В. Ломоносова, МФТИ и НИУ ВШЭ. Область научных интересов: автоматизация проектирования цифровой аппаратуры, верификация и тестирование.

Alexander Sergeevich KAMKIN is a leading researcher at the Software Engineering Department of ISP RAS, leading researcher at the Heterogeneous Computing Systems research lab of Plekhanov RUE. He is also a lecturer at MSU, MIPT, and HSE. His research interests include digital hardware design automation, verification and testing. Alexander has a PhD degree in Physics and Mathematics.

Михаил Михайлович ЧУПИЛКО – кандидат физико-математических наук, старший научный сотрудник отдела технологий программирования ИСП РАН, старший научный сотрудник научной лаборатории «Гетерогенные компьютерные системы» РЭУ им. Г.В. Плеханова. Область научных интересов: автоматизация проектирования цифровой аппаратуры, верификация и тестирование.

Mikhail Mikhaylovich CHUPILKO is a senior researcher at the Software Engineering Department of ISP RAS, senior researcher at the Heterogeneous Computing Systems research lab of Plekhanov RUE. His research interests include digital hardware design automation, verification and testing. Mikhail has a PhD degree in Physics and Mathematics.

Михаил Сергеевич ЛЕБЕДЕВ – научный сотрудник ИСП РАН, старший научный сотрудник научной лаборатории «Гетерогенные компьютерные системы» РЭУ им. Г.В. Плеханова; преподает в НИУ ВШЭ. Область научных интересов: высокоуровневый синтез, методы исследования проектных альтернатив, методы оптимизации, машинное обучение, цифровая аппаратура, методы верификации цифровой аппаратуры.

Mikhail Sergeevich LEBEDEV is a researcher at ISP RAS and the “Heterogeneous computer systems” laboratory of Plekhanov RUE. Research interests: high-level synthesis, design space exploration methods, neural networks, digital hardware, hardware verification.

Сергей Александрович СМОЛОВ – научный сотрудник отдела технологий программирования ИСП РАН, старший научный сотрудник научной лаборатории «Гетерогенные компьютерные системы» РЭУ им. Г.В. Плеханова. Область научных интересов: автоматизация проектирования цифровой аппаратуры, верификация и тестирование.

Sergey Aleksandrovich SMOLOV is a researcher at the Software Engineering Department of Ivannikov Institute for System Programming of the Russian Academy of Sciences (ISP RAS), senior researcher at the Heterogeneous Computing Systems research lab of Plekhanov RUE. His research interests include digital hardware design automation, verification and testing.

Георги ГАЙДАДЖИЕВ – профессор кафедры инновационных компьютерных архитектур Гронингенского университета и почетный приглашенный профессор Имперского колледжа Лондона, заведующий научной лаборатории «Гетерогенные компьютерные системы» РЭУ им. Г.В. Плеханова. Область научных интересов: архитектура и микроархитектура вычислительных систем, реконфигурируемые и гетерогенные вычисления, распределенные системы памяти, инструменты и методологии проектирования.

Georgi GAYDADJIEV received his M.Sc. (1996) in Electrical Engineering and Ph.D. (2007) in Computer Engineering from Delft University of Technology. He is currently a full professor in Innovative Computer Architectures at the University of Groningen and an honorary visiting professor at Imperial College, head of the “Heterogeneous computer systems” laboratory of Plekhanov RUE. His research interests include: Computer (System) Architecture and Micro-architecture, Reconfigurable and Heterogeneous Computing, Distributed and Advanced Memory Systems, Design tools and Methodologies, Low-Power Architectures, Architectural Support for Compilers and Runtime Systems.

DOI: 10.15514/ISPRAS-2022-34(5)-2

Практика и перспективы применения открытых и собственных программных решений в маршруте верификации систем на кристалле

[1,2]*А.В. Гаращенко, ORCID: 0000-0002-5147-6186 <agarashchenko@elvees.com>*
[1,3]*Д.С. Лашина, ORCID: 0000-0003-1252-228X <dlashina@elvees.com>*
[1,2]*С.А. Никитин, ORCID: 0000-0001-7307-5976 <sanikitin@elvees.com>*
[1]*А.В. Николаев, ORCID: 0000-0001-5141-2521 <anikolaev@elvees.com>*
[1,2]*Е.А. Прокопьев, ORCID: 0000-0002-7362-061X <eprokopev@elvees.com>*
[1]*Ф.М. Путря, ORCID: 0000-0002-3127-291X <fputrya@elvees.com>*
[1,2]*Б.Н. Цыренжапов, ORCID: 0000-0002-1212-0275 <btsyrenzhapov@elvees.com>*
[1]*АО НПЦ «ЭЛВИС»,*
124460, Россия, Москва, Зеленоград, ул. Конструктора Лукина, д. 14, стр. 142
[2] *Национальный исследовательский университет «МИЭТ»*
124498, Россия, Москва, Зеленоград, Площадь Шокина, д. 1
[3]*Национальный исследовательский ядерный университет «МИФИ»,*
115409, Россия, Москва, Каширское ш., 31

Аннотация. В статье рассматривается опыт и оценивается возможность применения свободного, открытого и собственного САПР в маршруте верификации СнК со степенью интеграции в миллиарды транзисторов, изначально базирующемся на коммерческих пакетах от "большой тройки". Предлагается подход к оценке пригодности конкретного САПР для заданного этапа маршрута верификации, основанный на формальном описании этапа и требований к средствам автоматизации для выбранного этапа. Приводятся собственные решения, внедренные в компании, являющиеся альтернативой коммерческим решениям или уникальными разработками. На основе предложенного подхода проведен анализ существующих средств автоматизации с точки зрения применимости в маршруте верификации современных СнК.

Ключевые слова: функциональная верификация; маршрут; система на кристалле; средства автоматизации; САПР; ПО с открытым исходным кодом; свободное ПО

Для цитирования: Гаращенко А.В., Лашина Д.С., Никитин С. А., Николаев А. В., Прокопьев Е. А., Путря Ф. М., Цыренжапов Б.Н. Практика и перспективы применения открытых и собственных программных решений в маршруте верификации систем на кристалле. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 23-42. DOI: 10.15514/ISPRAS-2022-34(5)-2

The practice and prospects of using open and proprietary software solutions in the verification route of SoC

[1,2]*A.V. Garashchenko, ORCID: 0000-0002-5147-6186 <agarashchenko@elvees.com>*
[1,3]*D.S. Lashina, ORCID: 0000-0003-1252-228X <dlashina@elvees.com>*
[1,2]*S.A. Nikitin, ORCID: 0000-0001-7307-5976 <sanikitin@elvees.com>*
[1]*A.V. Nikolaev, ORCID: 0000-0001-5141-2521 <anikolaev@elvees.com>*
[1,2]*E.A. Prokopev, ORCID: 0000-0002-7362-061X <eprokopev@elvees.com>*
[1]*F.M. Putrya, ORCID: 0000-0002-3127-291X <fputrya@elvees.com>*
[1,2]*B.N. Tsyrenzhapov, ORCID: 0000-0002-1212-0275 <btsyrenzhapov@elvees.com>*

[1] *ELVEES Research and Development Center,*
14, stroenie 14, Konstruktora Lukina st., Zelenograd, Moscow, 124460, Russia
[2] *National Research University of Electronic Technology (MIET)*
1, Shokin Square, Zelenograd, Moscow, Russia, 124498
[3] *National Research Nuclear University «MEPhI»*
115409, Russian Federation, Moscow, Kashirskoe shosse, 31

Abstract. This article discusses the experience and evaluates the capability of using free, open and internal proprietary EDA tools in the billon gates SoC verification flow initially based on the commercial EDA of the “Big 3”. Article suggests an approach to assessing the suitability of a particular EDA tools for a certain stage of the verification flow based on a formal stage description and requirements for the tool. It presents our internal tools implemented in the company, which are used as an alternative to commercial tools or as unique solutions. Based on the proposed approach, the analysis of the existing automation tools from the point of view of the applicability in the SoC verification flow was carried out.

Keywords: functional verification; verification route; SoC; system-on-a-chip; automation tools; ECAD; EDA; open source software; free software

For citation: Garashchenko A.V., Lashina D.S., Nikitin S. A., Nikolaev A. V., Prokopev E. A., Putrya F. M., Tsyrenzhapov B.N. The practice and prospects of using open and proprietary software solutions in the SoC verification flow. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 23-42 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-2

1. Введение

Функциональная верификация является частью маршрута верификации СнК, пронизывающая его от архитектуры до изготовления опытных образцов. Верификация является одной из самых затратных стадий разработки. При этом критическим является как время инженеров, необходимое на разработку и отладку окружений с тестами, так и время работы серверного парка и специализированной аппаратуры, необходимое на итеративные запуски моделирования и формальной верификации. По этой причине наличие специализированных САПР и аппаратного обеспечения, а также автоматизация процессов являются критическими в маршруте верификации.

Маршрут разработки (и, в частности, верификации) СнК с большой степенью интеграции, применяемый в подавляющем большинстве центров проектирования, сильно зависит от пакетов коммерческих САПР и аппаратуры, поставляемых "большой тройкой" (Cadence, Synopsys, Siemens (Mentor)).

Вопрос диверсификации применяемых в маршруте разработки СнК САПР неоднократно ставился на открытых площадках [1] и в рамках коллективов компаний. Однако ответ на вопрос о необходимости замены большинства применяемых САПР на собственные и открытые решения связан с оценкой экономической целесообразности. Проблемой альтернативных САПР является качество и ограниченный набор поддерживаемых свойств, негативно влияющих на сроки разработки и риски получения нерабочей микросхемы. Сроки

и затраты на разработку и поддержку альтернативных решений требуемого качества также оказываются крайне высокими.

Однако при сохранении в маршруте верификации стержня из САПР большой тройки, в ряде компаний с целью диверсификации и повышения эффективности процесса разработки все же разрабатываются собственные решения в области автоматизации, совместимые с маршрутом, построенным на коммерческих САПР. Кроме того, развиваются и открытые САПР. Преимущественно такие САПР применяются в академической среде, однако есть и примеры интереса к открытым продуктам со стороны больших коммерческих игроков, поддерживающих открытые решения. В таких начинаниях отметились компании типа Google [2], Siemens. Открытые решения также вызывают интерес у небольших стартапов и коллективов, занимающихся разработкой в ПЛИС, для которых задача верификации тоже стоит, хоть и менее актуальна, чем для заказной разработки.

С одной стороны, интерес к открытому ПО со стороны крупных игроков даёт толчок в развитии открытых решений, с другой стороны, остаются риски, что проекты рано или поздно все равно уйдут в коммерческий сегмент и закроются. Поэтому ситуацию с открытыми инструментами важно анализировать на постоянной основе (свежий пример – закрытие сайта SysWIP). Далее в статье детализируется маршрут верификации и проводится анализ доступных инструментов, решающих задачи автоматизации на каждом из этапов.

2. Этапы маршрута верификации

Для оценки применимости САПР из различных источников в маршруте верификации стоит разбить маршрут верификации на этапы, для каждого из которых будет характерен свой подход к автоматизации и свои нишевые САПР. Перечень этапов разработки СнК, в которых с различной степенью вовлечённости задействованы верификаторы, представлен ниже:

- разработка стратегии верификации;
- разработка верификационного плана;
- планирование и отслеживание статуса работ;
- управление исходными кодами проекта;
- разработка эталонных моделей блоков и системы;
- разработка или поиск верификационных компонент (VIP);
- разработка верификационных окружений;
- разработка тестовых сценариев (уровень СФ-блоков);
- линт и семантический анализ;
- формальная верификация RTL;
- генерация тестов для полной проверки автомата или перебора сценариев использования СФ-блока или системы;
- разработка тестовых сценариев (уровень подсистем и системный уровень);
- проверка RTL прикладным кодом;
- моделирование RTL и списков цепей (netlist);
- отладка RTL с использованием результатов моделирования;
- отладка окружения и тестов;
- пазработка тестовых сценариев с возможностью запуска на различных этапах маршрута разработки СнК;
- анализ производительности системы, поиск узких мест в архитектуре и реализации системы;
- автоматизация запуска регрессий и построения отчётов о статусе верификации;
- контроль достигнутого качества верификации;
- ускорение моделирования RTL; эмуляция;

- прототипирование СФ-блоков или системы;
- запуск тестов на опытных образцах ИС (bring-up);
- валидация ИС;
- поддержка пользователей в процессе эксплуатации микросхемы.

С формальной точки зрения линт и семантический анализ являются задачей разработчиков, передающих RTL на верификацию. Однако качество данного этапа напрямую связано с трудозатратами верификаторов на отладку полученного RTL, поскольку без линта и семантических проверок RTL может содержать большое число механических и логических ошибок, отнимающих время на отладку на первоначальном этапе запуска тестов. Проблемы метастабильности при переходе доменов тактового сигнала и сигнала сброса находятся за рамками цифровой симуляции и без применения специальных инструментов могут быть пропущены при выпуске СнК. Примером инструмента решающего задачу описанных выше проверок перед передачей RTL на верификацию является SpyGlass [3].

Прототипирование и стадии, следующие за tape-out микросхемы, напрямую не относятся к верификации, однако часто требуют привлечения специалистов, занимавшихся верификацией. Вопросы оптимизации маршрута, касающиеся данных стадий, рассмотрены в разделе, посвящённом созданию сценариев с возможностью запуска на различных этапах маршрута разработки СнК.

Стадии, напрямую касаюшиеся маршрута верификации, с различной степенью подробности будут рассмотрены далее в статье.

3. Способ сравнения САПР для этапа маршрута верификации на основе матрицы требований

Для каждого этапа верификации предлагается создание его формального описания, обязательно включающего перечень входных данных для этапа, описание действий, выполняемых над входными данными и описание формируемых по окончании этапа выходных данных. Данное описание используется для выделения перечня действий, которые требуется автоматизировать. Необходимость автоматизации обосновывается требованием соблюдения сроков и экономических затрат, отводимых на выполнение этапа верификации при достижении требуемого качества получаемых результатов.

Исходя из полученного перечня действий, подлежащих автоматизации, формируется перечень требований к САПР. Требования формулируются таким образом, что ответ на вопрос о соответствии оцениваемого САПР требованию может быть только бинарный (либо удовлетворяет, либо нет). Требования могут иметь приоритеты, определяющие степень важности для выбора САПР. Требование может быть принципиально необходимо для выполнения действия, либо необходимо для достижения лучших экономических показателей. В последнем случае указывается степень влияния на экономические показатели в баллах. Перечень требований и их ранжирование производится по методу экспертных оценок. Для получения более объективной оценки работа должна выполняться экспертами из нескольких компаний, представителей отрасли. На основе перечня требований проводится сравнение существующих коммерческих, свободных, открытых САПР и инструментов, созданных внутри компании. Для оценки применимости САПР в маршруте верификации выполнятся проверка закрытия всех принципиально необходимых свойств и оценивается полученная сумма баллов по всем закрытым требованиям для упрощения сравнения различных вариантов САПР.

При выборе САПР, а особенно в случае оценки открытого и свободного ПО, важно учитывать и факторы, связанные с доступностью и поддержкой инструмента в процессе эксплуатации. Требуется оценить объем вовлечённой в разработку инструмента команды, возможность прямого контакта с представителями команды разработки, интенсивность выпуска релизов,

дату выхода последнего релиза, число потребителей продукта. Данная информация критически важна для понимания, что использование продукта не приведёт к появлению блокирующих проблем в процессе работы над проектом (например, связанных, с ошибками в ПО САПР, которые не могут быть исправлены в короткие сроки). Приведённый набор требований по умолчанию должен входить в матрицы сравнения для выбора САПР.

В последующих разделах приводятся примеры описания этапов маршрута верификации и требований к применяемым на них САПР. Сравниваются известные САПР с точки зрения применимости в маршруте верификации, а также приводятся примеры использования сторонних и собственных решений, на различных этапах верификации.

4. Пример сравнения САПР для моделирования RTL модели на базе этапа моделирования RTL и списка цепей

4.1. Краткое описание этапа верификации

В табл. 1 представлено краткое описание этапа верификации.

Табл. 1. Описание этапа верификации
Table 1. Stage of verification description

Входные данные	Действия	Выходные данные
• RTL или нетлист (Verilog, VHDL, SV) • описание задержек (SDF) • описание доменов питания (UPF) • верификационное окружение • тесты • (verilog, VHDL, SV, UVM [4], C++) • скрипты управления моделированием	• моделирование цифровой схемы	• логи моделирования • временные диаграммы (например, VCD) • трассы транзакций и событий • дополнительная база данных для отладки • база данных покрытия (кода и функциональное)

4.2 Сравнение доступных САПР с учётом требования к этапу верификации

Сравнение доступных САПР представлено в табл. 2.

*Табл. 2. Сравнение доступных САПР (X/V/Q = XCELIUM, VCS, QUESTA) * - обязательные требования*
*Table. 2. Compare of available EDA (X/V/Q = XCELIUM, VCS, QUESTA) * - mandatory requirement*

Требования	Приоритет /баллы	BIG 3	Другие коммер-ческие (МИР)	Другие коммер-ческие (РФ)	Свободное	Открытое	Собственное
Verilog	*	X/V/Q		ereme x [5]		Icarus Verilator	
VHDL	*	X/V/Q		ereme x			
SystemC	*	X/V/Q				Verilator	

SystemVerilog (HW)	*	X/V/Q				Verilator	
SystemVerilog (VER)	*	X/V/Q					
SDF	*	X/V/Q					
UPF	*	X/V/Q					
Смешанное моделирование нескольких языков	*	X/V/Q		eremex			
нет ограничения по объёму проектов	*	X/V/Q					
Сбор покрытия кода	*	X/V/Q					
Сбор функционального покрытия	*	X/V/Q					
Сбор трасс транзакций	90	X/V/Q					
Скриптовое управление моделированием	80	X/V/Q					
Скорость моделирования позволяет достичь килогерцового эквивалента для верхнего уровня проектов СнК	100	X/V/Q				Verilator	

В таблице приведены только наиболее важные требования к САПР для демонстрации формы самой таблицы. В последующих разделах для экономии места таблицы тоже могут быть сокращёнными или опускаться.

4.3 Наличие открытых и собственных решений

Существует несколько открытых проектов цифровых симуляторов. Наиболее известные Verilator [6] и Icarus [7]. Данные проекты активно используются академическим сообществом. Verilator применяется и в коммерческих разработках для ускорения моделирования (однако при этом теряется детализация моделирования, например, потеря возможности хранить 4 состояния). Для отдельных СФ-блоков, особенно элементов вычислительного конвейера или алгоритмических ускорителей Verilator может применяться

с достаточной степенью эффективности при условии решения проблемы сбора покрытия альтернативными методами (например, по трассам).

Однако особенностью окружений проектов СнК является активное использование верификационных СФ-блоков (чаще всего имеющих API UVM) для проверки периферийных интерфейсов, число и сложность которых растёт с каждым проектом. Применение доменов частот и питания создаёт дополнительные ситуации, отладка которых слабо закрывается возможностями указанных симуляторов. Проект современной СнК, собираемый из IP разных поставщиков в большинстве случаев оказывается смешанным дизайном, содержащим код на нескольких HDL и указанными симуляторами не может быть промоделирован. Поэтому пока можно говорить только о нишевом использовании открытых инструментов типа Verilator, но для проектов СнК замены коммерческим симуляторам на данный момент нет.

5. Стратегия и планирование

Разработка стратегии требует инструмента, обеспечивающего многопользовательское редактирование документов. Решения для этой задачи широко представлены на рынке коммерческих и свободных инструментов. Типовой пример Confluence [8] Направления планирования и отслеживания задач закрываются инструментами с функциональностью MSProject [9] и Jira. Аналоги упомянутых инструментов широко представлены на рынке, в том числе имеются и свободные.

Задача создания верификационного плана является уже более критической. ПО для создания плана должно позволять трассировать свойства от ТЗ и спецификации до тестов и сценариев, которые должны быть исполнены в процессе моделирования. Для поддержания плана в актуальном состоянии важна интерактивность. Изменение ТЗ и спецификации должно сразу прослеживаться в редакторе плана. Описание тестов и покрываемые свойства должны быть привязаны к реальным тестам, регрессии и инструментам анализа покрытия. Поставщики САПР для верификации предоставляют подобные средства создания плана и контроля его исполнения. Как правило, они тесно интегрированы с САПР моделирования и запуска регрессий. Примером такого инструмента является VManager [10] от Cadence (в этом также году произошёл переход на платформу Verisium, объединяющей все инструменты верификации и внедряющей инструменты ИИ для оптимизации запуска и отладки регрессий [11]). Полноценных открытых альтернатив, сочетающих в себе весь перечисленный функционал, на данный момент нет.

6. Управление исходными кодами проекта

В настоящее время наиболее популярные системы контроля версий это Git [12] и SVN [13]. Позволяют сохранять историю изменений файлов исходных кодов, откатывать изменения, разграничивать права доступа к исходным кодам. Git в дополнении к вышеупомянутому позволяет настраивать политику доступа (раздельный доступ на чтение модификацию и т.д.), работать с отдельными ответвлениями, не затрагивая основной репозиторий, проводить минимальные обязательные проверки перед обновлением файлов репозитория, ставить метки на определённые срезы репозитория. Также существует множество коммерческих проектов вроде Perforce [14], Mercurial [15], предоставляющие расширенный функционал, вроде масштабируемости в Perforce, или большей скорости в Mercurial.

Проекты современных СнК разрабатываются несколькими командами, которые могут быть разнесены по разным локациям. Проект СнК разбивается на множество подпроектов, например, проекты СФ-блоков, эталонные модели, инструментальные средства разработки. Данные проекты могут иметь собственные репозитории, причём для большой компании может иметь место применение различных СКВ (систем контроля версий) для подпроектов. Проект всей системы собирается из множества подпроектов и, соответственно, репозиториев

подпроектов, в которых идут регулярные правки. Актуальным является наличие системы контроля исходных кодов, обеспечивающей стабилизацию проекта на всех уровнях иерархии и с учётом всех возможных зависимостей, включая инструментальное ПО, версии САПР и библиотек. Требуется регулярная фиксация иерархического среза проекта с возможностью гибкой настройки критериев для фиксации такого среза.

С учетом вышеперечисленного в нашей копании было принято решение о разработке проприетарной надстройки над системами git и svn, базирующейся на языке python и xml-описаниях зависимостейю Данный инструмент позволяет в одном проекте совместно использовать не только git и svn репозитории, но и задавать внешние зависимости от инструментария, например, компиляторов, библиотек или САПР, для которых используется собственный подход к хранению версий продуктов и выпуска релизов.

7. Разработка эталонных моделей, верификционных окружений и VIP

Высокоуровневая модель разрабатывается на этапе архитектурного проектирования и впоследствии может использоваться при верификации в качестве эталонной. Существуют коммерческие решения для построения виртуальных прототипов и моделей, однако базируются они в первую очередь на SystemC [16] и TLM [17] в качестве стандарта взаимодействия компонент. Разработка SystemC и C моделей возможна и без использования стороннего САПР (предполагается, что модели применяемых в проекте СФ-блоков доступны). Коммерческие решения позволяют создавать модели СнК быстрее и имеют интеграцию с САПР для моделирования, однако данные свойства не являются критическими для маршрута разработки моделей.

Существенные затраты времени уходят на разработку и отладку верификационного окружения. Одни из наиболее затратных компонент с точки зрения разработки – верификационные СФ-блоки (VIP). На разработку VIP для сложного интерфейса требуется несколько человеко-лет, а таких VIP в проекте требуется десятки типов. Компании из "большой тройки" предлагают свои VIP для решения данной проблемы. На рынке VIP имеются компании не входящие большую тройку, например [18], есть открытые решения VIP [19]. В стенах нашей компании разработана библиотека VIP для наиболее часто используемых накристальных интерфейсов [20] и ряда периферийных интерфейсов.

Для ускорения разработки верификационных окружений существует продукты, предоставляющие возможность генерации UVM окружений (есть коммерческие решения, например, SYSVIP от Cadence) и фреймворки с открытым исходным кодом для построения верификационных окружений и генерации тестов. В рамках верификации крупных СнК окружениям необходимо удовлетворять таким требованиям, как возможность использования VIP третьих сторон, скорость моделирования, удобство разработки и отладки большим коллективом разработчиков, а также удобство повторного использования кода между окружениями на различных уровнях иерархии проекта.

Подход к тестированию с использованием SystemVerilog и UVM является наиболее эффективным, но обладает повышенным порогом вхождения. В данный момент активно развивается подход к написанию тестов при помощи языка высокого уровня Python с использованием фреймворка CocoTb, обеспечивающего взаимодействие с RTL моделью [21] CocoTB применим для верификации автономных IP-блоков, однако данный фреймворк обладает целым рядом недостатков. К ним относятся сложность переноса тестов на системный уровень, прототипы и чипы, низкая скорость моделирования, невозможность интеграции UVM VIP третьих сторон, недостаток средств отладки.

Существуют фрейморки, ориентированные на оптимизацию процесса сборки окружений и запуска тестов. VUnit – фреймворк с открытым исходным кодом, созданный для целей автоматизированного тестирования RTL-кода, написанного на языке Verilog. Данный

фреймворк не заменяет, а призван дополнить традиционные методы тестирования посредством автоматизации. VUnit нацелен на минимизацию временных затрат на верификацию благодаря автоматическому обнаружению иерархии модулей, порядка компиляции, а также порядка включения библиотек. Скорость разработки повышается за счёт инкрементальной компиляции и возможности декомпозиции тестбенчей. Качество проектов при этом повышается, позволяя запустить большие регрессионные наборы на сервере непрерывной интеграции[22]

SVUnit – Фреймворк, аналогичный предыдущему, однако обладающий расширенной функциональностью и поддержкой языка SystemVerilog [23]. Оба упомянутых фреймворка не решают задачи собственно генерации окружений

Для ускорения процесса создания окружения в компании был разработан генератор UVM окружений. В качестве входных данных принимает метаинформацию о проекте (например, число и тип интерфейсов, карту памяти устройства, IPXACT описание блока) и тип проекта (автономное окружение, окружение коммутатора, окружение системы). Генерируемое окружение удовлетворяет перечисленным выше требованиям и содержит дополнительные свойства, связанные с переносимостью тестов между стадиями проектирования (см. далее).

8. Разработка тестовых сценариев, запускаемых на различных стадиях маршрута разработки СнК

Разработка и отладка тестов является одним из основных этапов и занимает большую часть времени инженеров верификаторов. Типовая ситуация для компаний, занимающихся разработкой СнК – наличие нескольких проектов СнК, как разработанных ранее, так и проектируемых в параллель. Очевидным решением с точки зрения оптимизации является повторное использование кода тестов, написанных для блоков, повторно применяемых в проекте. Однако, чтобы обеспечить максимальную простоту повторного использования, требуется соблюдение целого ряда правил написания и хранения кода тестов, позволяющую абстрагировать код теста СФ-блока от особенностей конкретного проекта, таких как архитектура процессора, особенности организации управления системой тактирования, контроллером прерываний, памятью и т.п.

От проекта к проекту конфигурация и версия СФ-блока, выполняющего схожую функцию (например, контроллер SPI, блок таймеров), может меняться. При этом большая часть тестовых сценариев, необходимых для проверки интеграции блока с точки зрения алгоритма теста остаётся одинаковой, при этом прямой перенос кода алгоритма с проекта на проект невозможен из-за отличий в регистровой модели СФ-блока.

Ещё одним направлением для повторного использования кода тестовых сценариев является необходимость их воспроизведения на различных уровнях иерархии проекта и на различных этапах разработки СнК. Например, перенос теста с автономного окружения на системный уровень для проверки интеграции блока или на прототип для проверки корректности функционирования прототипа.

Один из подходов к решению описанных проблем является стандарт PSS. Стандарт PSS (Portable Test and Stimulus Standard) позволяет описывать сценарии на самом высоком уровне, которые являются источником при генерации тестов под конкретный проект. В PSS также можно описать все ограничения блока и то, какие он действия выполняет. Все это описывается математическими графами, которые более понятны и удобны при написании сценария теста. PSS описывает намерение теста, которое должно одинаково интерпретироваться на всех уровнях (системы и подсистемы) и во всех других проектах, где будет использоваться данный блок [24].

Для генерации конечных тестов используется инструмент Perspec. Perspec – инструмент для автоматической генерации теста, который основан на модели. В Perspec можно составить

варианты использования (сценарии) и рандомизировать их. Также Perspec позволяет собирать данные о сценарном покрытии [25]. Обычно покрытие анализируются после запуска симуляций тестов. Механизм perspec позволяет собрать данные о необходимом покрытии до того, как верификатор переходит к генерации и запуску теста. Данный функционал помогает сократить число итераций при написании теста, который должен полностью покрывать заданную группу сценариев.

Однако для обеспечения портируемости сценариев, созданных с применением PSS, требуется наличие дополнительного кода, обеспечивающего запуск сценария на каждой из платформ. Кроме того, переход на создание кода на чистом PSS делает код тестов радикально отличающимися от кода целевого ПО, что усложняет повторное использования кода между стадиями верификации и разработки ПО. Негативным эффектом тут будет снижение скорости получения работоспособного ПО на микросхеме после её изготовления и затруднения при использовании кода прикладного ПО на этапе верификации.

В компании была предложена другая концепция создания переносимых тестовых сценариев на базе создания абстрактных классов тестовых сценариев для различных типов устройств. Обеспечить наилучшие условия разработки портируемых тестов позволяет кроссплатформенная библиотека тестовых функций libcpptest, содержащая драйвера устройств. Такие драйвера специалист создаёт и разрабатывает самостоятельно. При этом драйвер обращается к регистрам и макросам доступа к регистрам, которые генерируются исходя из IPXACT [26] описания блока, а также использует ядро упомянутой библиотеки для абстрагирования типовых операций взаимодействия с системой.

Библиотека тестовых сценариев предполагает создание сценариев на высоком уровне абстракции для групп устройств. Такой подход позволяет расширять код сценариев и настраивать их запуск на системном уровне независимо от конкретной реализации СФ-блока. Для обеспечения такой возможности все СФ-блоки делятся на группы, для каждой из которых создается абстрактный драйвер, содержащий в себе задекларированные методы, которые имеются во всех похожих блоках. Впоследствии эти методы реализуются в конкретных драйверах. К примеру, можно рассмотреть абстрактный блок таймера. Каждый таймер предназначен для отчета времени, которое в него заранее заложено. Поэтому один из сценариев, который можно написать для каждого таймера - это фиксация времени, что отчитывается от запуска таймера и до выработки прерывания (или иного события). Для абстрагирования таймера можно выделить следующие методы общие для каждого таймера: конфигурация и запуск таймера, остановка таймера, сброс таймера, сброс прерывания. Исходя из этих методов можно написать следующий алгоритм универсального теста таймеров:

- конфигурация таймера (параметры должны задаваться случайно);
- запуск таймера;
- фиксация времени перед запуском таймера;
- ожидание срабатывания таймера;
- фиксация времени останова таймера;
- сброс и остановка таймера;
- сравнение полученного интервала с эталонным значением.

При этом абстрактный сценарий может требовать определения как функций в драйвере конкретного СФ-блока, применяемого в системе, так и функций, зависящих от проекта системы или окружения; в приведенном примере это функция фиксации времени.

При конструировании тестовых сценариев для верификации подсистемы или системы достаточно важную роль играет разработка интеграционных тестов, таких как тесты регистров, памяти, прерываний и создание комплексных тестов, проверяющих корректность подключения устройств и их функционирование, в том числе при параллельной работе устройств в составе системы. Каждый из тестов имеет свои цели, объекты и способы реализации, и вместе формируют достаточно большой фронт работ, поэтому для повышения эффективности труда инженера-верификатора были разработаны специальные инструменты, позволяющие автоматизировать процесс создания драйверов, окружений и их сборку [27], [28] на базе платформенного подхода. С использованием упомянутых инструментов и библиотек для ускорения процесса создания тестов системного уровня был разработан специальный инструмент – конструктор типовых тестовых сценариев для проверки регистров, доступа к памяти, прерываний, комплексных сценариев проверки параллельной работы блоков.

Главной целью комплексного теста является проверка корректности взаимодействия СФ-блоков между собой путём одновременного или поочерёдного запуска набора тестовых сценариев верификации СФ-блоков и их взаимодействия в системе. Для его конструирования в библиотеке libcpptest создан ряд виртуальных классов, позволяющих разрабатывать потоки тестовых сценариев и конструировать из них комплексные тесты системного уровня [29]. При этом процесс разработки потоков тестового сценария и управление тестом на верхнем уровне позволяет разнести эти процессы. Так, интсрумент автоматизированного распределения памяти для тестовых сценариев разрабатывается независимо от тестовых потоков.

Ядро библиотеки libcpptest обеспечивает переносимость тестовых сценариев на автономные, системные окружения прототипы и на чипы и делает возможным применение всех упомянутых ранее подходов на всех этапах проектирования чипа.

Ядро библиотеки подразумевает разделение теста на два уровня: низкий – уровень драйверов, и высокий – уровень тестового приложения. В ядро введены 4 абстрактных класса, предоставляющими уровень абстракции от аппаратуры для взаимодействия с контроллером прерываний, доступа к памяти и регистрам (включая как прямой, так и backdoor доступ), а также механизмы отладочной печати.

9. Формальная верификация

На полный разбор средств и подходов к формальной верификации цифровых схем не хватит объёма одной статьи. В рамках формата текущей статьи считаем важным отметить, что изначально наиболее интенсивно инструменты для формальной верификации развивались в академической среде и существует большое число открытых и академических проектов разной степени проработанности, посвящённых формальной верификации. Изначально коммерческие инструменты были в роли догоняющих, однако за последнее десятилетие коммерческие инструменты сделали большой шаг вперёд и такие продукты как Cadence Jasper Gold от Cadence позволяют выполнять проверку свойств для дизайнов в миллионы транзисторов предлагают широкий набор дополнительных возможностей и адаптивный выбор решателей. Порог входа для использования коммерческих решений для верификаторов ниже чем у открытых альтернатив, что делает их использование выгодней с точки зрения сроков разработки. Однако в направлении открытых альтернатив работа ведётся достаточно интенсивно, в частности есть проекты формальной верификации на основе инструмента Yosys [30].

В рамках задачи поиска решений в области формальный верификации инженерами нашей компании был повторен результат, описанный в [31]. Как и в первоисточнике задача формальной верификации решалась для арифметических блоков АЛУ процессора. В итоге задача формальной верификации была успешно решена. Недостатком инструментов

оказалась не полная поддержка *Verilog* 2001 (*IEEE* 1364-2001), приведенная к необходимости адаптации проверяемого RTL под ограничения применяемого инструмент.

10. Генерация тестов для полной проверки автомата или перебора сценариев использования СФ-блока или системы

В рамках функциональной верификации СнК на этапе разработки тестов выделяют две глобальные задачи: полная проверка автоматов СФ-блоков и перебор сценариев их использования. Основными метриками оценки степени проверки выступают: значения тестового покрытия кода, блоков, выражений и сигналов, полученные при помощи САПР. Для максимизации полноты покрытий, а также проверки отсутствия ошибок и клинчей в архитектуре, реализации СФ-блоков и протоколов их взаимодействия помимо ручного написания применяются подходы автоматизированной генерации тестов.

Часто для автоматизации формирования тестов применяют методы на основе математических моделей [32] и методы на основе псевдослучайного дополнения шаблонов [33]. Генерация тестовых сценариев на основе методов с применением математических моделей явным образом не привязана к языкам программирования или верификации. Например, технология UniTESK и поддерживающие ее инструменты основаны на использовании моделей конечных автоматов [34]. Однако для построения математических моделей такого класса существует проблема экстрагирования пространства состояний автомата из формальных спецификаций и исходного кода RTL, что обусловлено высокой комбинаторной сложностью как отдельных СФ-блоков, так и системы в целом. Комбинирование различных алгоритмов генерации тестов и математических моделей СФ-блоков позволяет создавать как ожидаемые, так и критические ситуации для системы. На практике используют комбинации [35] в зависимости от имеющихся ресурсов.

Помимо этого, для генерации тестов применяются высокоуровневые модели блоков и подсистем, среди средств реализации которых используются следующие языки: C++, SystemC/TLM, SystemVerilog. Для проектов типа процессоров или акселераторов наличие полноценной высокоуровневой модели всего блока или устройства является обязательным, поскольку только при наличии такой модели возможно построить полноценный маршрут верификации на базе случайных и направленных тестов. Сгенерированный код представляет собой программные последовательности, написанные на языках ассемблера или типичных языках программирования встраиваемых систем (C/C++), компилируемых в базис целевой архитектуры системы и исполняемые на RTL. Данный подход активно используется для проверки вычислительных ядер, блоков программного управления, подсистемы памяти, предсказаний переходов и т.п. Код запускается на высокоуровневой модели, рассчитываются значения регистров и памяти, создаются их дампы, которые в дальнейшем используются при сравнении значений, полученных из соответствующего запуска кода на RTL.

Отдельной проблемой верификации на основе методов генерации тестов является продолжительность их моделирования, т.к. формирование тестов относится к классу задач, сложность (количество сгенерированных тестов) которых растет экспоненциально с ростом числа входных данных. Следовательно, для конструирования меньшего количества тестов, а соответственно и уменьшения общего времени моделирования, необходимо использовать такие методы искусственного ограничения пространства математической модели (например, вершин и ребер графа), как обобщение и унификация её состояний. Т.е. рассматривать структурные единицы модели на более абстрактном уровне без учета конкретных значений. При этом основной акцент генерации смещается на стрессовые для системы ситуации, приводящие её к различным блокировкам.

Для решения проблемы генерации тестов на основе описания автомата состояний и переходов существуют коммерческие инструменты, такие как Cadence Perspec,

базирующийся на стандарте PSS. Имеются также альтернативные инструменты, упомянутые выше. В рамках задачи верификации ядер DSP и CPU в компании были разработаны собственные инструменты генерации тестов для проверки автоматов datapath и кэшей данных процессоров.

11. Отладка RTL, тестовых окружений и тестов

Описание этапа верификации представлено в табл. 4.

Табл. 3. Описание этапа верификации
Table. 3. Description of verification stage

Входные данные	Действия	Выходные данные
• логи моделирования • временные диаграммы • трассы транзакций, инструкций и событий • дополнительная база данных для отладки • база данных покрытия	• анализ временных диаграмм и состояний регистров и памяти • анализ исходных кодов и схемы с привязкой к состоянию в заданный момент • анализ логов моделирования • анализ трасс транзакций и событий • сравнение временных диаграмм • сравнение трасс • анализ состояния динамических объектов окружения • пошаговая отладка RTL и тестбенча • подключения программного отладчика для анализа состояния процессора с привязкой к исходным кодам программы • поиск причины ошибки • составление инструкции для воспроизведения ошибки	• описание ситуации, в которой возникла ошибка, описание ожидаемого и фактического поведения • описание предполагаемой причины ошибки • инструкции и шаги для воспроизведения ошибок

В контексте маршрута верификации проектов СнК этап отладки, включающий применение большого количества САПР, один из самых ресурсоемких и продолжительных. На данном этапе итерационно производится поиск, анализ и локализация проблем, которые связаны не только с ошибками в самом RTL, но и с ошибками в окружении и тестах, поэтому задачу отладки формально возможно разделить на две смежных и часто пересекающихся по используемому инструментарию задачи: отладка RTL и отладка тестовых окружений и тестов.

Существуют разные подходы к отладке RTL, большинство из которых связаны с использованием результатов моделирования. Каждая итерация этапа часто требует перезапуска процесса моделирования с новыми параметрами, что резко ограничивает временные ресурсы. Основными признаками возникающих проблем корректности работы системы могут выступать: пропуск или расхождения в трассах транзакций и исполнения программ; дополнительная печать в логах прохождения тестов; недостаточное тестовое или функциональное покрытие; сработавшие assertions. На практике для минимизации времени моделирования RTL в тестовые последовательности добавляются блоки самопроверки и дополнительной печати, которые позволяют фиксировать ошибки, не покрытые при помощи assertions за счёт сравнения с эталонной моделью. Также разрабатываются специальные трассировочные блоки, встраиваемые в RTL и позволяющие в динамике «на лету» логировать дополнительную информацию, например, программную трассу вычислительных ядер.

Соответственно, первоочередным после фиксации проблемной ситуации и визуального просмотра логов при наличии эталонной модели проверяемого СФ-блока является сравнение

трасс. Часто простого сравнения недостаточно, инженерам-верификаторам необходимо разрабатывать дополнительные «парсеры», средства сведения логов и интеллектуального сравнения трасс, учитывающие недетерменированные ситуации с точки зрения несходимости моделей. Производится попытка локализации проблемы и осуществляется поиск возможных причин. После фиксации ситуации встаёт вопрос её воспроизводимости с возможностью восстановления контекста из сохранных SNAPSHOT для «длинных» тестовых последовательностей. Если на данной итерации не удаётся установить причину проблемы, необходимо произвести более глубокий анализ с использованием временных диаграмм, дампов конфигурационных регистров и памяти.

В зависимости от принципа и автомата работы СФ-блока к поиску проблем возможно зайти с разных сторон. Однако с точки зрения требований, предъявляемых к САПР, необходима конвергенция программных решений: визуализации временных диаграмм; менеджеров значений регистров и памяти; трассировщика связей блоков с возможностью отслеживания передачи значений шин и сигналов; редактирования исходного RTL-кода. Помимо этого, при моделировании необходима возможность фиксации, перебора и рандомизации задержек на интерфейсах. Также удобным и порой важным инструментом для воспроизведения и анализа проблем, возникающих при моделировании списка цепей, на RTL является средство сведения и сравнения временных диаграмм, полученных на разных запусках симуляции.

Для отладки тестовых окружений и тестовых воздействий применяется как подход, описанный выше, так и специфические именно для окружения шаги. Для окружений и тестов чаще применяются методы отладки, характерные для отладки ПО, возможно использование встроенных средств в связке с САПР: GDB, пошаговой отладки кода, анализа состояний объектов и переменных, дизассемблера, анализа секций объектного файла и т.д.

В части инструментов анализа трасс, в частности анализа трасс процессорных ядер или трасс транзакций в маршруте отдела верификации, используются как коммерческие решения, так и собственные решения, не уступающие коммерческим по функциональности. Однако в части анализа временных диаграмм в связке со схемой и исходными кодами альтернативы коммерческим инструментам в маршруте нет, кроме того, в инструментах типа SimVision от Cadence есть целый ряд дополнительных функций, ускоряющих процесс отладки.

12. Анализ производительности системы, поиск узких мест в архитектуре и реализации системы

Для обеспечения сходимости проекта СнК по критерию эффективности на реальных задачах весь маршрут разработки СнК, начиная с архитектурного проектирования и заканчивая квалификационными тестами топологического списка цепей, должен быть ориентирован на раннюю локализацию проблем с производительностью и подтверждение потребительских характеристик СнК перед её запуском на фабрику (tapeout). На начальных этапах для оценки производительности используются высокоуровневые модели, как правило TLM SystemC, которые в первом приближении позволяют оценить эффективность предлагаемой архитектуры для решения целевых задач. Для решения задач построения высокоуровневых моделей существует ряд коммерческих САПР, в частности, Cadence предлагает инструмент Helium Virtual and Hybrid Studio. Ранее отмечалось, что именно в части моделирования высокоуровневых прототипов критической зависимости от коммерческого САПР нет, хотя есть зависимость от поставщиков моделей IP. Однако экосистему для удобного построения моделей большая тройка предлагает в достаточной мере проработанную. В работе [36] подробно приводятся причины расхождения результатов оценки производительности на TLM и RTL моделях и необходимость оценки характеристик СнК на RTL. В этом плане преимуществом коммерческих решений является и возможность смешанного моделирования с использованием симуляторов и эмуляторов для уточнения параметров модели в процессе

детализации отдельных компонент, а также встроенных инструментов для анализа узких мест.

В компании был разработан ряд инструментов, базирующийся на подходе к формированию трасс транзакций и событий в процессе моделирования RTL, их анализа, автоматической проверки метрик производительности и визуализации результатов [37]. Инструменты разработанные внутри компании позволяют закрыть задачу анализа производительности системы на фазе разработки RTL, за счёт применения её декомпозиции и построения инфраструктуры для запуска тестов производительности и анализа их результатов на автономных окружениях вычислительных ядер, окружениях подсистем памяти, коммутаторов, а также подтверждения параметров на выборочном наборе тестов на RTL системного уровня [38].

13. Автоматизация запуска регрессий и контроль достигнутого качества верификации

Действия, требующие автоматизации со стороны комплекса управления регрессиями и контроля качества верификации, приведены в перечне ниже:

- запуск теста вручную с заданными параметрами;
- запуск регрессии по событию (расписание, при изменении исходных кодов в репозитории);
- запуск квалификационных тестов для кандидата в релиз (RTL, среда верификации или обе сущности);
- распараллеливание запуска регрессии с использованием доступных вычислительных ресурсов;
- хранение результатов запусков для всех перечисленных возможных вариантов запусков тестов;
- визуализация результатов запусков пакетов тестов для текущего среза проекта или релизов проекта;
- возможность просмотра текущего состояния верификации в режиме онлайн;
- оповещение заинтересованных лиц о резких изменениях состояния верификации проекта;
- накопление данных по достигнутому покрытию кода и функциональному покрытию;
- визуализация достигнутого статуса по покрытию по результатам прохождения пакета тестов;
- визуализация достигнутого статуса по покрытию по результатам накопления данных от серии запусков пакетов тестов;
- визуализация данных о закрытии позиций верификационного плана по результатам запуска пакета тестов;
- построение отчетов о статусе верификации по расписанию по результатам запуска регрессии;
- накопление и визуализация дополнительных метрик тестов для отладки регрессии (время исполнения теста, число предупреждений и ошибок в логах и т.п.)’
- возможность настройки отчета о статусе верификации и метриках тестов (определение отображаемых параметров и способа их визуализации для ускорения анализа статуса регрессии, возможность настройки разных типов отчетов, например, для инженеров и для менеджеров).

Коммерческие инструменты, например, Vmanager либо самостоятельно решают все перечисленные задачи, либо предоставляет интерфейс для python-скриптов для настройки

формируемых отчётов или поведения регрессии проекта. Однако, построение системы запуска регрессии и получения отчётности возможно и на открытых и свободных инструментах.

В процессе верификации проекта уровня СнК может возникнуть множество проблем, влияющих на общий ход событий и, таким образом, спровоцировать изменения в планах, сформированных изначально. В случае возникновения проблем в процессе верификации скорость локализации проблемы представляет наибольшую важность, поскольку в случае позднего обнаружения проблемы зачастую бывает затруднительно понять, с какого момента началась проблема, а главное — после каких изменений. В связи с этим важен контроль достигнутого качества верификации на текущий момент с возможностью отслеживания исторических изменений.

Наблюдаемые метрики и характеристики теста определяются ответственными за верификацию проекта и механизмы сбора, как правило, носят проприетарный характер. В компании такой механизм реализован в рамках ПО Project Compiler (ProjectCompiler - проприетарное ПО АО НПЦ "ЭЛВИС" впервые упомянуто в [39]). Экстрагированная из лог-файлов информация хранится в базе данных в специальном формате. В нашем случае информация хранится в формате JSON в базе данных на СУБД PostgreSQL, поскольку данная СУБД предоставляет гибкие возможности работы с JSON-структурами внутри полей. Формат JSON выбран исходя из соображения минимизации трудозатрат по добавлению новых метрик и характеристик теста: хранение в виде JSON позволит добавить новую информацию, не прибегая к модификации DDL-описания сущности базы данных. Оптимизация процесса контроля статуса верификации проектов достигается за счёт автоматизированного непрерывного анализа состояния регрессий.

В настоящее время для построения полноценной системы контроля качества верификации на всем её протяжении какого-либо одного инструмента недостаточно. Используют комбинации и проприетарных и свободных инструментов, таких как Vmanager, Gitlab, Jenkins, Grafana и практически всегда есть необходимость использования собственного проприетарного ПО, объединяющего перечисленные инструменты в маршруте и закрывающего отдельные функции, специфичные для внутреннего маршрута компании.

14. Ускорение моделирования RTL. Эмуляция

Компании из "большой тройки" поставляют специализированную аппаратуру для ускорения моделирования [40] и прототипирования цифровых схем. Без использования специализированной аппаратуры, позволяющей запускать большие объёмы тестов и прикладного кода (вплоть до ОС) представить себе верификацию СБИС СнК с должной степенью проверки сегодня нельзя. К сожалению, аналогов эмуляторов нет даже за пределами "большой тройки". Инструменты для прототипирования представлены более разнообразно, есть даже примеры их создания компаниями в России [41], однако на текущий момент остаётся зависимость от элементной базы и САПР для синтеза в ПЛИС.

15. Выводы

В маршруте функциональной верификации современных СнК полностью отказаться от использования САПР большой тройки в данный момент невозможно. Так, симуляторы и средства отладки только от большой тройки закрывают ряд критических свойств (например в части поддержки стандартов SystemVerilog, SDF, UPF, возможностей по сбору функционального покрытия и покрытия кода). Аппаратные средства ускорения моделирования в данный момент не имеют аналогов. Усугубляется ситуация тем, что многие средства автоматизации выстраиваются на основе возможностей, предоставляемых симулятором и, таким образом привязаны к решениям от большой тройки.

Тем не менее с большинством этапов верификации, за пределами обозначенных выше, ситуация обстоит не так критично, а это почти десяток инструментов и утилит помимо симулятора. Существуют как альтернативные коммерческие предложения, так и открытые решения либо, внутренние решения компаний, занимающихся разработкой СнК (в частности упомянутые в данной статье решения на примере одной компании). Использование альтернативных решений позволяет как диверсифицировать маршрут верификации в части зависимости от поставщиков САПР и поддержки получаемой от них, так и получить новые свойства, позволяющие повысить эффективность решения задач верификации.

В частности, в нашей компании были разработаны следующие собственные решения, закрывающие часть этапов верификации:

- генераторы верификационных окружений;
- библиотека собственных VIP (UVM SystemVerilog);
- инструменты для анализа производительности сетей на кристалле;
- методология разработки тестов, переносимых между стадиями верификации;
- система конфигурируемых генераторов тестов для вычислительных ядер;
- универсальное средство сборки и запуска окружений и тестов.

В компании успешно применяется открытое и свободное ПО для следующих этапов:

- система управления исходными кодами и выпуска релизов;
- система запуска регрессионного тестирования и анализа результатов таких запусков;
- отдельные VIP.

Также был получен опыт применения открытого и свободного ПО для следующих этапов:

- создание стратегии верификации;
- планирование работ;
- формальная верификация (арифметические блоки);
- создание верификационного плана;
- отслеживание задач и ошибок.

ПО из последнего списка было впоследствии заменено на коммерческое, поскольку последнее позволяет решать описанные задачи с большей эффективностью. Однако опыт показал принципиальную возможность использования свободного и открытого ПО на данных этапах, и в случае необходимости, с учётом готовности пойти на определённое снижение эффективности работы, возможно использовать и альтернативные решения.

Упомянутые в статье альтернативные (как открытые, так и собственные) решения не являются продуктами, сопоставимыми по потребительским качествам с коммерческими. Т.е. имеется задел по САПР для многих этапов, однако по каждому из перечисленных этапов доведение САПР до продуктового качества потребует усилий команд разработчиков.

Для более оптимального построения работы по разработке и поиску альтернативного САПР предлагается использовать описанный в статье подход по выработке и ранжированию требований к альтернативным решениям. При этом к составлению перечня требований должны привлекаться специалисты из большинства центров проектирования, являющихся потребителями САПР.

Кроме того, центры проектирования СнК могут и сами в ряде случаев являться поставщиками инструментов разработки, поскольку имеют внутренние решения, поддрживающие этапы маршрута проектирования.

Список литературы / References

[1] Обзор конференции МЭС-2018 / Overview of the MES-2018 conference. Available at: http://www.mes-conference.ru/index.php?prev=mesPrev&yp=2018&mpd=mpd_srv#sv, accessed 20.10.2022 (in Russian).

[2] Build Custom Silicon with Google. Available at: https://developers.google.com/silicon, accessed 20.10.2022.
[3] SpyGlass. Available at: www.syposys.com, accessed 20.10.2022
[4] Harshitha N.B., Praveen Kumar Y.G., Kurian M.Z. An Introduction to Universal Verification Methodology for the digital design of Integrated circuits (IC's): A Review. In Proc. of the International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 1710-1713.
[5] Компания ЭРЕМЕКС. Официальный сайт / EREMEX Company. Official site. Available at: https://www.eremex.ru/, accessed 20.10.2022 (in Russian).
[6] Verilator. Available at: https://www.veripool.org/verilator/, accessed 20.10.2022.
[7] Icarus simulator. Available at: http://iverilog.icarus.com/, accessed 20.10.2022.
[8] Atlassian. Available at: https://www.atlassian.com, accessed 27.10.2022
[9] MSProject. Available at: https://www.microsoft.com/, accessed 27.10.2022
[10] Zhang D. Smarter Verification Management with vManager Platform. In Proc. of the DVCon China, 2021, 40 p.
[11] Cadence Revolutionizes Verification Productivity with the Verisium AI-Driven Verification Platform. Available at: https://www.businesswire.com/news/home/20220913005378/en/Cadence-Revolutionizes-Verification-Productivity-with-the-Verisium-AI-Driven-Verification-Platform, accessed 27.10.2022.
[12] Git. Available at: https://git-scm.com, accessed: 21.11.2022.
[13] Apache Subversion. Available at: https://subversion.apache.org, accessed: 21.11.2022.
[14] Perforce Software. Available at: www.perforce.com, accessed 27.10.22.
[15] Mercurial. Available at: www.mercurial-scm.org, accessed 27.10.22.
[16] SystemC. Available at: https://systemc.org/, access 27.10.22.
[17] OSCI TLM-2.0 Language Reference Manual. Available at: https://www.accellera.org/images/downloads/standards/systemc/TLM_2_0_LRM.pdf, accessed 27.10.2022.
[18] SmartDV. Available at: https://www.smart-dv.com/, accessed 27.10.22.
[19] OpenCores. Available at: https://opencores.org/, accessed 27.10.22.
[20] AMBA Specifications. Available at: https://www.arm.com/architecture/system-architectures/amba/amba-specifications, accessed 27.10.22.
[21] What is cocotb. Available at: https://docs.cocotb.org/en/stable/index.html accessed 27.10.22.
[22] What is Vunit. Available at: https://vunit.github.io/about.html, accessed 31.10.2022.
[23] SVUnit User Guide. Available at: http://agilesoc.com/open-source-projects/svunit/svunit-user-guide, accessed 27.10.2022.
[24] Balance M. Automating test from IP to SoC levels with portable stimulus, Available at: https://www.techdesignforums.com/practice/technique/automating-test-from-ip-to-soc-levels-with-portable-stimulus/, accessed 28.10.2022.
[25] Cadence Perspec System Verifier Supports New Accellera Portable Test and Stimulus Specification 1.0, Available at: https://www.design-reuse.com/news/44368/cadence-perspec-system-verifier-accellera-portable-test-and-stimulus-specification-1-0.html, accessed 31.10.2022
[26] Никитин С.А., Николаев А.В и др. Автоматизация маршрута функциональной верификации на основе стандарта IP-XACT. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), вып. 4, 2020 г., стр. 90-94 / Nikitin S.A. Nikolaev A.V. et al. Automation of functional verification flow based on IP-XACT standard. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 4, 2020, pp. 90-94 (in Russian).
[27] Жезлов К.А., Колбасов Я.С. и др. Автоматизация процесса создания тестовых окружений, обеспечивающая сквозной маршрут разработки, верификации и исследования СФ-блоков и СнК. Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС), вып 2, 2016 г., стр. 46-53 / Zhezlov K.A., Kolbasov Y.S. et al. Automation of verification environments development process providing a through design flow for design, verification and research of IP-BLOCKS and SOC. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 2, 2016, pp. 46-53 (in Russian).
[28] Фролова С.Е., Путря Ф.М. Создание многофункциональной и мультипроектной платформы прототипирования на базе комплекта HAPS компании Synopsys. Часть 2. Электроника: наука, технология, бизнес, вып. 5, 2019, стр. 120-125 / Frolova S.E., Putrya F.M.. Creation of multi-functional and multi-project prototyping platform based on synopsys haps kit. Part 2. Electronics: Science Technology Business, вып. 5, 2019, pp. 120-125 (in Russian).

[29] Головина Е.К., Макеева М.А. и др. Метод создания и отладки комплексных тестов для функциональной верификации СнК, ориентированный на их повторное использование на всех этапах проектирования. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), вып. 2, 2014 г., стр. 45-50 / Golovina E.K., Makeeva et al. The method of building and debugging complex tests for the SoC functional verification, oriented on their reuse at all stages of the development. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 2, 2014, pp. 45-50 (in Russian).

[30] Symbiyosys. Available at: https://symbiyosys.readthedocs.io/en/latest/index.html#symbiyosys-sby-documentation, accessed 20.10.2022.

[31] David M. Russinoff. Formal Verification of Floating-Point RTL with ACL2, 2019. Available at: https://www.cl.cam.ac.uk/~jrh13/spisa19/paper_06.pdf, accessed 27.10.2022

[32] Garashchenko A.V., Putrya F.M. et al. Automatic Test Generation Methodology for Verification of a Cache Memory Based on the Graph Model of Cache Hierarchy. In Proc. of the IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 2019, pp. 1876-1879.

[33] Chupilko M., Kamkin A. et al. MicroTESK: Specification-Based Tool for Constructing Test Program Generators. Lecture Notes in Computer Science, vol. 10629, 2017, pp. 217-220.

[34] Иванников В.П., Камкин А.С. и др. Применение технологии UniTesK для функционального тестирования моделей аппаратного обеспечения. Препринт ИСП РАН, вып. 8, 2005 г., стр. 1-26 / Ivannikov V.P., Kamkin A.S. Application of UniTesK technology for functional testing of hardware models. ISP RAS preprints, issue 8, 2005, pp. 1-26 (in Russin).

[35] Garashchenko A., Nikolaev A.V. et al. System of Combined Specialized Test Generators for the New Generation of VLIW DSP Processors with Elcore50 Architecture. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 2, 2019, pp. 2-7.

[36] Путря Ф.М. Особенности верификации современных систем на кристалле. Труды международной конференции «Микроэлектроника-2015», 2016 г., стр. 49-57 / Putrya F.M. System on chip verification issues. In Proc. of the International Conference «Microelectronics-2015», 2016, pp. 49-57 (in Russian).

[37] Жезлов К.А., Колбасов Я.С. и др. Этапы маршрута верификации и валидации системы, ориентированные на получение модели СнК, способной гарантированно исполнять целевые задачи и алгоритмы с заданными характеристиками. Вопросы радиоэлектроники, вып. 8. 2018 г., стр 64-72. / Zhezlov K.A., Kolbasov Ya. S. et al. Steps of verification and validation route aimed at obtaining the SOC model able to perform target tasks and algorithms with specified characteristics. Issues of Radio Electronics, issue 8. 2018, pp. 64-72 (in Russian).

[38] Жезлов К.А. Методика автоматизированного анализа производительности коммутационных сред с учетом структуры СнК и прикладных. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), вып. 3, 2020 г., стр. 94-99 / Zhezlov K.A. Methodology of automated performance analysis of SOC interconnect subsystems, according to SOC structure and application. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 3, 2020, pp. 94-99 (in Russian).

[39] Жезлов К.А., Колбасов Я.С. и др. Автоматизация процесса создания тестовых окружений, обеспечивающая сквозной маршрут разработки, верификации и исследования СФ-блоков и СнК. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), вып. 2, 2016 г., стр. 46-53. / Zhezlov K.A., Kolbasov Ya.S. et al. Automation of verification environments development process providing a through design flow for design, verification and research of IP-blocks and SOC. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 2, 2016, pp. 46-53 (in Russian).

[40] Flaherty N. Cadence boosts its emulation and verification systems, Available at: https://www.eenewseurope.com/en/cadence-boosts-its-emulation-and-verification-systems/, accessed 20.10.2022

[41] Юрлин С.В., Бычков И.Н. Прототипирование на основе ПЛИС для верификации многоядерных микропроцессоров. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), вып. 4, 2014 г., стр. 45-50. / Yurlin S.V., Bychkov I.N. FPGA prototyping for functional verification of multi-core processors. Problems of development of advanced micro-and nanoelectronic systems (MES), issue 4, 2014, pp. 45-50 (in Russian).

Информация об авторах / Information about authors

Антон Витальевич ГАРАЩЕНКО – кандидат технических наук, ведущий инженер АО НПЦ "ЭЛВИС", доцент института Института системной и программной инженерии и информационных технологий НИУ МИЭТ. Сфера научных интересов: автоматизация верификации гетерогенных многоядерных СнК.

Anton Vitalievich GARASHCHENKO – Candidate of Technical Sciences, Staff Engineer at JSC ELVEES RnD Center, Assistant Professor of SSEIT NRU MIET. Research interests: automation of heterogeneous SoC verification.

Дарья Сергеевна ЛАШИНА – техник АО НПЦ "ЭЛВИС", студентка магистратуры кафедры ИИКС НИЯУ МИФИ. Сфера научных интересов: функциональная верификация СнК, информационные технологии.

Daria Sergeevna LASHINA – Technician at ELVEES RnD Center, JSC, Master's Student of ICIS MEPHI. Research interests: functional verification, computer science.

Святослав Александрович НИКИТИН – старший инженер АО НПЦ "ЭЛВИС", аспирант кафедры ПКИМС НИУ МИЭТ. Сфера научных интересов: автоматизация верификации цифровых СФ-блоков и СнК, машинное обучение.

Svyatoslav Aleksandrovich NIKITIN – Senior Engineer of ELVEES RnD Center, JSC, postgraduate student of DICD NRU MIET. Research interests: Digital IP and SoC verification automation, Machine Learning.

Артём Валерьевич НИКОЛАЕВ – кандидат технических наук, руководитель группы автоматизации верификации СФ-блоков и СнК. Сфера научных интересов: автоматизация верификации цифровых СФ-блоков и СнК.

Artyom Valerievich NIKOLAEV – Candidate of Technical Sciences, Head of the Verification Automation Group. Research interests: digital IP and SoC verification automation.

Евгений Андреевич ПРОКОПЬЕВ – техник АО НПЦ "ЭЛВИС", студент бакалавриата кафедры МПСУ НИУ МИЭТ. Сфера научных интересов: функциональная верификация СнК, информационные технологии.

Evgeny Andreevich PROKOPEV – technician at ELVEES RnD Center, JSC, student-bachelor of NRU MIET. Research interests: functional verification, computer science.

Федор Михайлович ПУТРЯ – кандидат технических наук, начальник отдела верификации. Сфера научных интересов: Многоядерные системы на кристалле (СнК), функциональная верификация, автоматизация, анализ производительности.

Fedor Mikhailovich PUTRYA – Candidate of Technical Sciences, Head of the Verification Department. Research interests: Multicore SoC, functional verification, automation, performance analysis.

Булат Намсараевич ЦЫРЕНЖАПОВ – техник АО НПЦ "ЭЛВИС", студент бакалавриата кафедры НМСТ НИУ МИЭТ. Сфера научных интересов: функциональная верификация СнК, информационные технологии.

Bulat Namsaraevich TSYRENZHAPOV – technician at ELVEES RnD Center, JSC, student-bachelor of NRU MIET. Research interests: functional verification, computer science.

DOI: 10.15514/ISPRAS-2022-34(5)-3

Метод восстановления протокольных автоматов по бинарному коду

И.В. Шарков, ORCID: 0000-0002-9767-142X <sharkov@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Реверс-инжиниринг сетевых протоколов широко применяется в задачах анализа безопасности сетевых программ. Задача реинжиниринга протокола состоит из восстановления форматов данных и менее изученного вопроса – восстановления реализованного в программе протокольного автомата, а краеугольным камнем является отсутствие формально понятия состояния протокола. В настоящее время сложились два основных подхода к восстановлению автоматов сетевых протоколов, базирующиеся на использовании различной исходной информации: на основе анализа записанных сетевых трасс и путем анализа бинарного кода программы, реализующего исследуемый протокол. В статье предлагается метод восстановления протокольных автоматов на основе анализа бинарного кода трасс программ в процессе их выполнения. Первой целью работы является описание математической модели протокольного автомата и метода ее проецирования на бинарный код приложения. Вторая цель – описание концепции понятия состояния протокола и правил, указывающих на выполнение переходов, с использованием некоторых «глобальных» объектов трассы программы. Третья преследует способ уточнения протокола с помощью фаззинга в памяти процесса (in-memory fuzzing) с использованием «плавающей» точки запуска порождающего сервера для контроля состояний и управления переходами. Наконец, в статье очерчена схема разработанного набора инструментов и показаны эксперименты по его использованию с реальным VPN-клиентом, подтверждающие состоятельность подхода.

Ключевые слова: сетевой протокол; протокольный автомат; состояние; бинарный код; фаззинг

Для цитирования: Шарков И.В. Метод восстановления протокольных автоматов по бинарному коду. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 43-62. DOI: 10.15514/ISPRAS-2022-34(5)-3

Protocol automata recovery method using binary code

I.V. Sharkov, ORCID: 0000-0002-9767-142X <sharkov@ispras.ru>
Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. Security analysis of network programs includes set of reverse engineering tasks of network protocols. Data formats restoring and implemented protocol automaton are the previous task issues. Unlike quite researched problem of formats restoring where there are lots of scientist’s papers, finding out the protocol's automaton program implementation looks like terra incognita and the cornerstone is a protocol state description currently undefined. There are two general ways to retrieve the implemented protocol automaton: an analysis of the network traces and looking into binary trace of the target application. This article offers a second one method. The first aim of the paper is the way to describe a mathematical model of a protocol automaton and a method for projecting it onto an executing application binary code. The second is concept of the protocol state definition and a principle to detect the states transitions based on some "global" binary trace objects, are described. Thirdly, there is suggested a protocol automaton precising manner by in-memory fuzzing based on a "floating" fork-server to manage states transitions. Finally, developed toolset's scheme and experiments on its using with a real VPN client, are shown.

Keywords: network protocol; protocol automata; state; binary code; fuzzing
For citation: Sharkov I.V. Protocol automata recovery method using binary code. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 43-62 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-3

1. Введение

Реверс-инжиниринг сетевых протоколов применяется в задачах анализа безопасности сетевых программ. Решения данной задачи позволяет обнаруживать отклонения исследуемого протокола от эталона, выявлять отличия, возникшие в ходе реализации различных программных средств, сравнивать версионные изменения, оценивать совместимость программ, обнаруживать проблемные места.

Восстановление спецификаций недокументированных протоколов и протокольных автоматов путем анализа исходных текстов программ задача весьма трудоемкая, требующая значительного времени [1-3], кроме того, исходные тексты не всегда доступны. Это подталкивает исследователей выбирать другие способы получения информации, позволяющей восстанавливать спецификации протоколов и протокольные автоматы, и создавать автоматизированные средства для решения проблемы.

Пути решения данной задачи в условиях отсутствия доступа к исходным текстам (рис. 1) можно разделить на два направления: анализ сетевых трасс и исследование бинарного кода приложений. При этом следует отметить, что наличие сетевых трасс в ходе анализа бинарного кода играет вспомогательную роль.

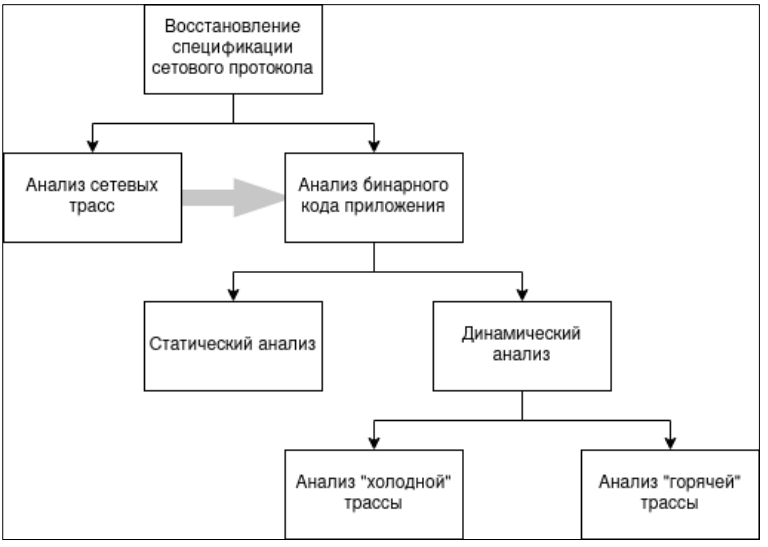


Рис. 1. Способы восстановления спецификаций протоколов и протокольных автоматов
Fig. 1. Ways for recovering protocol specifications and protocol automata

Использование записанных сетевых трасс в качестве начальных данных для восстановления протокольного автомата позволяет решать данную задачу в условиях, когда невозможно получить доступ к бинарному коду программы, при этом применяются, как аналитические подходы, так и автоматизированные алгоритмы, основанные на статистических методах.

Анализ бинарного кода приложения методом изучения записанных («холодных») трасс хорошо автоматизируется, позволяет применять технологии динамического символьного выполнения, восстанавливать алгоритмы.

Применение методов динамической инструментации программ позволяет исследовать «горячую» трассу, т. е. изучать бинарный код программы в ходе ее выполнения. Такой способ восстановления протокольного автомата из бинарного кода – это своего рода фаззинг, но в отличие от «классического» понимания фаззинга здесь основной задачей выступает

восстановление закономерностей и всех возможных последовательностей сообщений [1], а не подбор входных данных, вызывающих некорректное поведение программы (аварийное завершение, зависание и т. п.), другими словами, объектом изучения подобных исследований является не состояние программы, а состояние протокола. Дальнейшее повествование статьи посвящено именно этой проблематике.

Задача реинжиниринга спецификации протокола состоит из восстановления форматов данных и менее изученного вопроса – восстановления реализованного в программе протокольного автомата. Автомат связан с внутренним состоянием сетевой программы, переход из одного состояния в другое выполняется при получении пакета входных данных определенного формата. Для восстановления понимания работы программы необходимо уяснить, что именно определяет состояние протокола и по каким правилам происходят переходы. Кроме того, после решения указанной задачи необходимо выполнить тестирование на предмет определения насколько корректно и полно восстановлен автомат. В статье предлагаются методы, позволяющие автоматизировано решать эти задачи.

2. Обзор научно-технической информации

В настоящее время сложились два основных подхода к восстановлению протокольных автоматов сетевых протоколов, базирующиеся на использовании различных начальных данных: на основе анализа записанных сетевых трасс и путем анализа трассы выполнения программы, реализующей исследуемый протокол. Оба указанных способа имеют право на существование и позволяют решать поставленную задачу в различных начальных условиях.

Например, при осуществлении исследований проприетарных промышленных протоколов, используемых для автоматизации управления производственными процессами, объектов социальной или критической инфраструктуры в условиях отсутствия доступа к исполняемым файлам и исходным текстам решение задачи возможно только путем анализа записанных сетевых трасс.

В других случаях, когда в ходе решения задачи существует возможность запустить бинарный код, реализующий протокол, в контролируемом окружении, для исследования можно применять информацию, полученную в ходе анализа трасс выполнения программ или программно-аппаратных комплексов.

В обзорной статье венгерских исследователей [4], рассмотрены 39 инструментов и технологий реверс-инжиниринга сетевых протоколов и восстановления протокольных автоматов, информация о которых была представлена в свободном доступе в период с 2005 по 2017 гг. Среди них только 13 позволяют восстанавливать протокольные автоматы. Аналогичный обзор, представленный в источнике [5] в начале 2021 года, позволяет расширить список результатами, полученными после 2017 года, среди которых, в контексте задачи восстановления протокольных автоматов, можно выделить 2 работы [6, 7].

Наиболее свежие результаты были представлены в августе 2021 года в статье [2], при этом применяется метод восстановления форматов сообщений, описанный в работе [8] тех же авторов.

Обобщенные сведения о результатах указанных работ и возможностях созданных инструментов приведены в табл. 1.

Табл. 1. Инструменты восстановления протокольных автоматов
Table 1. List of protocol automata reverse engineering tools for network protocols

Название/авторы	Год	Практические результаты			Вход		Выход	
		Т. пр.	**Б. пр.**	**Др. пр.**	**СТ**	**ТВ**	**ФП**	**ПА**
GAPA [9]	2005	HTTP				✔	✔	✔
PEXT [3]	2007	FTP			✔	✔		✔

Prospex [10]	2009	SMTP, SIP	SMB	Agobot (C&C)	✔	✔	✔	✔
Xiao и др. [11]	2009	HTTP, FTP, SMTP				✔		✔
Trifilo и др. [12]	2009		TCP, ARP, DHCP, KAD		✔			✔
Antunes and Neves [13]	2009	FTP			✔			✔
ReverX [14]	2011	FTP			✔		✔	✔
Veritas [15]	2011	SMTP	PPLIVE, XUNLEI		✔			✔
Biprominer [16]	2011		XUNLEI, QQLive, SopCast		✔		✔	✔
Zhang и др. [17]	2012	HTTP, SNMP, ISAKMP			✔			✔
Netzob [18, 19]	2012	FTP, Samba	SMB	P2P? VoIP?	✔		✔	✔
AutoReEngine [20]	2013	FTP	DHCP		✔			✔
Laroche и др. [21]	2013	HTTP, FTP, SMTP, POP3	DNS, NetBIOS		✔		✔	✔
Meng и др. [22]	2014		TCP, ARP		✔			✔
PREUGI [6]	2017				✔			✔
Goo и др. [7]	2019	HTTP	DNS		✔		✔	✔
Gábor Székely и др. [2]	2021				✔		✔	✔

т.пр. – текстовые протоколы, **б.пр.** – бинарные протоколы, **др.пр.** – другие неизвестные протоколы, **СТ** – сетевая трасса, **ТВ** – трасса выполнения программы, **ФП** – формат протокола, **ПА** – протокольный автомат

Из приведенной таблицы видно, что в большинстве случаев восстановление протокольных автоматов основано на анализе ранее записанных сетевых трасс. Основная проблема при таком подходе – это восстановление форматов сообщений. Формат сообщения, это набор полей и их объединение в структуры и последовательности, а также информация о семантике полей – их роли в структуре сообщения. В условиях отсутствия знаний об алгоритме обработки сообщения восстановление информации о его составляющих элементах и их семантике с достаточной точностью является весьма сложной задачей. Для ее решения используются различные эвристические, статистические [15], грамматические [6, 17] алгоритмы, методы анализа последовательностей [23] и т.д. Наибольшие трудности, очевидно, возникают при анализе бинарных протоколов. Кроме того, очевидно, что в случаях применения кодирования или криптографических методов защиты передаваемых данных задача восстановления форматов сообщений из сетевой трассы становится, в лучшем случае, трудноразрешимой.

В подходе [2] для описания протокольного автомата используется автомат Мили, отличающийся от обычного конечного автомата тем, что для каждого перехода, инициированного входным сигналом, определен выходной сигнал. В предложенном методе применяются модифицированный алгоритм L_M+ [5] построения автомата по принципу обучения и алгоритм минимизации автомата [24]. Однако в данной работе не рассматриваются подходы к восстановлению форматов и классификации сообщений, предполагается, что эта задача уже решена.

Способы восстановления протокольных автоматов на основе записи и дальнейшего анализа трасс выполнения целевого бинарного кода в приведенной таблице датируются более чем десятилетней давностью. С точки зрения функциональных возможностей среди указанных работ наибольший интерес представляет Prospex [10]. В данном подходе для получения начальных данных требуется пара приложений клиент и сервер. Интересующие программы запускаются на выполнение в контролируемом окружении, при этом осуществляется анализ потока данных при обработке входных сообщений, в результате восстанавливаются форматы сообщений. Далее осуществляется анализ сетевых трасс, анализируются цепочки обмена данными, а сообщения классифицируются с использованием алгоритма анализа последовательностей [23], после чего в соответствии с восстановленными форматами вырабатываются обобщенные форматы. На последнем этапе, используя имеющиеся цепочки обмена сообщениями и обобщенные форматы, с помощью эвристических алгоритмов строится протокольный автомат, работа завершается минимизаций автомата с помощью алгоритма Exbar [25]. Ограничением в данной работе выступает отсутствие возможности работы с клиентскими приложениями. Кроме того, в открытом доступе не представлено исходных текстов инструмента Prospex, а исследователи более не публиковали какие-либо дополнительные результаты по данной тематике. Вместе с тем, предложенный подход выглядит перспективным и будет использован в качестве отправной точки для создания метода восстановления протокольных автоматов сетевых протоколов.

3. Математическая модель протокольного автомата

3.1 Общее описание

Существуют два подхода к формальному описанию процессов, функционирующих в режиме «запрос-ответ», с помощью математического аппарата конечных автоматов [26].

1) За основу берутся состояния (protocol based/protocolar). В данном случае в качестве вершин графа выступают состояния протокола, а ребра описывают обобщенные действия и задают функцию переходов из состояния в состояние, выполняемых под воздействием внешних сигналов. Этот способ описания позволяет строить абстрактные протокольные автоматы.
2) За основу берутся действия или поведение (behavior based/behavioral). В качестве вершин определяются действия системы, в результате выполнения которых вырабатывается сигнал, а ребра описывают этот сигнал и определяют функцию переходов. Такой подход позволяет описывать управляющие автоматы. Подобного рода автоматы показывают связь между состояниями программы и состояниями протокола.

В статье для описания протокольных автоматов используется первый поход, однако, стоит отметь, что используется аппарат не классических конечных автоматов, а, скорее, его расширение, допускающее использованием некоторого набора переменных, влияющих на поведение. Кроме того, необходимо обратить внимание, что этот подход применим для описания систем, осуществляющих взаимодействие в режиме реального времени, а не в отложенном режиме.

Рассмотрим основные понятия, необходимые для дальнейших рассуждений.

Под *системой* будем понимать программное средство осуществляющее взаимодействие с использованием сетевого интерфейса.

Протокол – это формально описанный абстрактный язык взаимодействия/коммуникации систем.

Системы будем называть *однородными*, если они могут осуществлять длительное результативное взаимодействие с использованием одного и того же протокола, т. е. системы при выполнении запросов получают релевантные ответы.

Передаваемые в рамках взаимодействия однородных систем данные формируются на основе грамматических и синтаксических правил протокола и называются *сообщениями*.

Входящее (входное) сообщение – это сообщение, поступившее в систему, исходящее (выходное) сообщение – отправленное системой.

В случаях, когда хотя бы одно сообщение зависит от одного или нескольких предыдущих входных и/или выходных, протокол называется протоколом чувствительным к состояниям (stateful-протокол), в противном случае – протоколом не чувствительным к состояниям (stateless-протокол).

В ходе проведения исследований в изученных материалах не было найдено определения понятия *состояние протокола*, которое можно однозначного формализовать и описать с использованием элементов математических теорий, в силу чего определим его следующим образом.

Состояние протокола – это состояние всех *глобальных объектов* (некоторых данных, значения которых определяются в процессе взаимодействия однородных систем и используются до завершения их коммуникации), оказывающих влияние на порядок следования и содержимое входных и выходных сообщений. Исходя из приведенного определения, stateless-протоколы не имеют глобальных объектов, чем и отличаются от stateful-протоколов.

Начальное состояние протокола – состояние, в котором система ожидает получения (серверная часть) или отправки (клиентская часть) первого сообщения.

Конечное (терминальное) состояние протокола – состояние, в котором невозможно дальнейшее получение входных и отправка выходных сообщений в рамках данного сеанса сетевого взаимодействия. Важно отметить, что в отдельных случаях конечное состояние может отсутствовать, например при реализациях промышленных протоколов, нацеленных на длительное непрерывное функционирование, например, в IoT-устройствах.

Смена состояния протокола осуществляется под воздействием входных сообщений (сигналов), при этом индикатором перехода в следующее состояние является отправка выходного сообщения или закрытие канала связи.

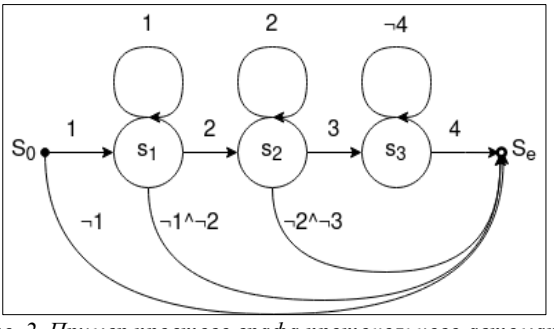


Рис. 2. Пример простого графа протокольного автомата
Fig. 2. Simple protocol automata graph sample

Протокольный автомат – это абстрактная математическая модель, предназначенная для формального описания протокола. Для наглядности протокольный автомат удобно представлять в виде ориентированного графа переходов, в качестве вершин которого

выступают состояния протокола, а ребра задают *функцию перехода* из состояния в состояние и помечаются входными и, при необходимости, выходными сигналами (рис. 2).

	1	*2*	*3*	*4*	*др.*
S_0	S_1	S_e	S_e	S_e	S_e
S_1	S_1	S_2	S_e	S_e	S_e
S_2	S_e	S_2	S_3	S_e	S_e
S_3	S_3	S_3	S_3	S_e	S_3
S_e	–	–	–	–	–

Рис. 3. Описание протокольного автомата с помощью таблицы
Fig. 3. Table representation of the defined early protocol automata

Кроме графического представления, используется табличная (матричная) запись автоматов. Каждой строке такой таблицы соответствует одно состояние, а столбцу входной сигнал. В ячейке на пересечении строки и столбца записывается следующее состояние и, если требуется, выходной сигнал. Табличное представление автомата, показанного на рис. 2, приведено ниже (рис. 3). Данный способ удобен с точки зрения программной реализации протокольного автомата.

Обобщая изложенное, зададим протокольный автомат *M* шестеркой объектов:

$$\mathrm{M} = (S, P, Q, f, s_0, s_e),$$

где:

S: $|S| < \infty$ – множество состояний протокола;

P: $|P| < \infty$ – множество входных сигналов;

Q: $|Q| < \infty$ – множество выходных сигналов;

f: $S \times P \times Q \to S$ – функция переходов из состояния в состояние;

$s_0 \in S$ – начальное состояние;

$s_e \in S$ – конечное состояние (в отдельных случаях конечное состояние может отсутствовать).

Множество состояний протокола *S*, а также множества входных *P* и выходных сигналов *Q* конечны. Функция f: $S \times P \times Q \to S$ осуществляет переход в следующее состояние, при этом оценивается текущее состояние протокола, анализируется входной сигнал и формируется выходной сигнал для ответной части. При этом для любого состояния протокола (кроме конечного) существует входной сигнал, переводящий его в другое, то же самое состояние или конечное состояние, и соответствующий этому выходной сигнал:

$$\forall\, s \in S : s \neq s_e, \exists\, p \in P, \exists\, q \in Q \; u \; \exists\, s' \in S : f(s,p,q) = s'.$$

3.2 Проецирование математической модели на бинарный код

Для построения протокольного автомата сетевого протокола компьютерной программы необходимо спроецировать построенную математическую модель на бинарный код, для чего требуется решить следующие задачи:

- определить множество состояний *S*, зависящее от некоторых глобальных объектов и стека вызовов, выделить начальное и конечное состояния (может отсутствовать);
- построить множества *P* и *Q* входных и выходных сообщений, соответственно, сформировать возможные пары *(p, q)* из множества $P \times Q$;
- построить функцию переходов;
- определить эквивалентные состояния и минимизировать протокольный автомат.

Как было определено ранее, состояние протокола характеризуется состоянием неких глобальных объектов, по значениях которых или их совокупности можно судить о различии состояний протокола и выполнении перехода из одного в другое. Таким образом, для

построения множества состояний необходимо определить набор глобальных объектов и обеспечить возможность контроля и сравнения их значений на различных этапах выполнения программы.

Выполнившийся алгоритм можно представить в виде линейно упорядоченного списка вызванных инструкций. Линейное представление бинарного кода позволяет определить зависимости одних данных от других. В частности, если на некотором шаге *i* полученные данные были каким-либо образом обработаны и сохранены (на стеке или в куче), а на каком-то из следующих шагов $j > i$ они были использованы в процессе обработки вновь полученных данных или подготовки выходного сообщения, тогда, очевидно, что новые данные зависят от полученных ранее при условии, что в интервале шагов *(i, j)* интересующие нас блоки данных в адресном пространстве не перезаписывались. В состав глобальных объектов введем множество S_{num} – порядковых номеров инструкций в трассе выполнения программы.

Реализация протокольных автоматов в программном коде может быть выполнена различными способами, в том числе крайне нестандартными. Вместе с тем, в ходе изучения исходных текстов сетевых программ, например, СУБД Postgres, FTP-сервера ioFTPD и др., а также, исходя из личного опыта разработки, были выделены четыре наиболее часто встречаемых. Рассмотрим их на примере протокольного автомата, представленного в виде графа на рис. 5.

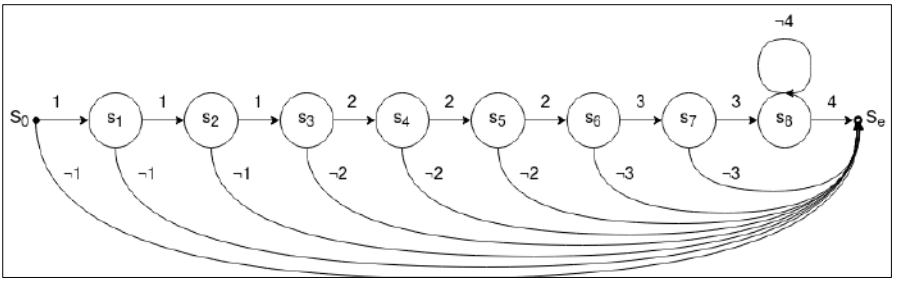


Рис. 5. Пример протокольного автомата для дальнейших рассуждений
Fig.5. Protocol automata sample for the further reasoning

1) Линейная последовательность вызовов функции приема и выполнения анализа входных сообщений, приводящая к конечному циклу обработки. С-подобный псевдокод приведен в листинге 1.

```
…
recv(…buf…);
if(buf != 1)cleanup_and_exit();
recv(…buf…);
if(buf != 1)cleanup_and_exit();
recv(…buf…);
if(buf != 1)cleanup_and_exit();
recv(…buf…);
if(buf != 2)cleanup_and_exit();
while(1) {
  recv(…buf…);
  if(buf == 4) cleanup_and_exit();
}
…
```

Листинг 1. С-подобный псевдокод линейного алгоритма
Listing 1. C-like pseudocode of the linear algorithm

В данном случае легко видеть, что смена состояния протокола будет равносильна смене адреса инструкции вызова функции получения входных данных, а в общем случае, если

для приема входных данных реализована отдельная функция, тогда различающимся состояниям соответствуют различающиеся стеки вызовов, завершающиеся вызовом системной функции `recv(…)`. Будем рассматривать это утверждение в качестве критерия, позволяющего судить о смене состояния в ходе анализа бинарного кода, а стек вызовов до вызова функции получения сообщения в качестве одного из глобальных объектов протокола.

2) Последовательный набор циклов (листинг 2).

```
int i = 0;
while(i<3) {
  recv(…buf…); i++;
  if(buf != 1) cleanup_and_exit();
}
i = 0;
while(i<3) {
  recv(…buf…); i++;
  if(buf != 2) cleanup_and_exit();
}
i = 0;
while(i<2) {
  recv(…buf…); i++;
  if(buf != 3) cleanup_and_exit();
}
while(1) {
  recv(…buf…);
  if(buf == 4) cleanup_and_exit();
}
```

Листинг 2. C-подобный псевдокод алгоритма последовательных наборов циклов
Listing 2. Algorithm C-like pseudocode of the sequential cycles

Применяя критерий смены состояния, сможем построить автомат, показанный на рис. 6. Нетрудно видеть, что изначальный автомат не будет эквивалентен полученному, но будет его частным случаем.

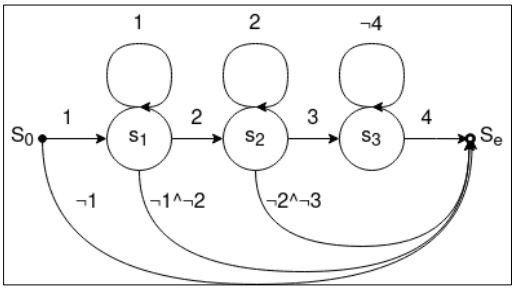


Рис. 6. Автомат, построенный по предложенному коду с использованием первого критерия
Fig. 6. Construction of protocol automata based on the code sample using the first criteria

С точки зрения состояний программы состояния протокола будут описываться значением переменной *i* на конкретном шаге алгоритма. Однако привязать эту переменную к обработке пакетов при анализе трассы выполнения весьма затруднительно, но, проследив количество выполнений одной и той же инструкции вызова функции получения входных данных, можно конкретизировать полученный автомат. Таким образом, в качестве одного из глобальных объектов протокола можно рассматривать счетчик количества выполнения инструкции вызова функции получения данных, сопровождающихся

последующей отправкой выходного сообщения и соответствующих одному и тому же стеку вызовов.

3) Единый цикл получения и анализа данных с блоком условных переходов (листинг 3).

```
prot_state cur_state = INIT_STATE;
while(curr_state != END_STATE) {
  recv(…buf…);
  switch(buf) {
    case 1: … break;
    case 2: … break;
    case 3: … break;
    case 4: … break;
    default: … break;
  }
}
```

Листинг 3. С-подобный псевдокод цикла с блоком условных переходов
Listing 3. C-like pseudocode of a single cycle with conditional branch's block

Очевидно, что, в данном случае, при обработке очередного пакета в зависимости от содержимого сообщений или вычисляемых на их основе значений переменных граф потока управления меняется.

Для различия сетевых сообщений одного протокола используют различные контрольные значения их содержимого: заголовки, команды, служебные данные и т. п., - назовем их ключевые поля.

Ключевые поля сообщений строго определены. Их обработка обеспечивает возможность выбирать конкретное направление развития алгоритма, формирует правила принятия решений о смене состояния протокола. В бинарном коде эта задача решается с помощью инструкций условных переходов, предназначенных для передачи управления одной из заданных инструкций. Таким образом, описанный ранее линейный список выполненных инструкций представляется в виде графа потока управления, вершинами которого выступают адреса инструкций, а ребра представляют собой передачу управления от одной инструкции к другой. Сообщения будут равны тогда и только тогда, когда графы потока управления, сформированные в результате работы алгоритма их обработки, равны.

Очевидно, что при различных значениях ключевых полей, после их сравнения с референтными значениями, во время выполнения инструкций условных переходов будут выбираться различные пути развития алгоритма обработки сообщений, в силу чего и сформированные графы потока управления будут различаться.

Другими словами, вектора, составленные из адресов пройденных базовых блоков, будут различными при различных состояниях протокола. Совокупность таких векторов можно ввести в состав глобальных объектов протокола.

4) Единый цикл получения и обработки данных, где в зависимости от полученного сообщения изменяется указатель на функцию обработчик (листинг 4).

```
typedef void* (*process_func)(void*);
typedef struct _state_ {
  byte id;
  process_func process_message;
} prot_state;
prot_state cur_state(INIT_STATE);
while(curr_state != END_STATE) {
  recv(…buf…);
```

```
    cur_state =

(prot_state*)state[cur_state].process_message(buf);
}
```

Листинг 4. С-подобный псевдокод цикла с изменяющейся функцией-обработчиком сообщения
Listing 4. C-like pseudocode of a single cycle with a variable processing function

В данном примере очевидно, что с использованием первого критерия все полученные сообщения будут отнесены к одному и тому же состоянию, однако состояние программы в ходе ее выполнения будет меняться, в частности переменная *cur_state*, значение которой зависит от обработанных входных данных, и указатель на вызываемую функцию-обработчик. Таким образом, можно утверждать, что если в ходе выполнения программы в процессе обработки очередного входного сообщения в паре:

$$\langle addr_{call\_instr}, addr_{callee\_fn} \rangle,$$

где:

$addr_{call\_instr}$ – адрес инструкции вызова;

$addr_{callee\_fn}$ – адрес вызываемой функции;

изменяется значение $addr_{callee\_fn}$ – адреса функции, при одном и том же $addr_{call\_instr}$ – адресе инструкции вызова, тогда можно утверждать, что протокол сменил состояние, т.е. совокупность глобальных объектов протокола можно дополнить парами указанного вида.

Обобщая изложенное, исходя из сформулированного определения состояния протокола, в качестве глобальных объектов протокола будем рассматривать сохраненные в адресных пространствах данные, доступные в рамках работы алгоритмов протокола, а также адреса инструкций, вызывающих функции (системные вызовы) получения сообщений и адреса самих функций.

Введем обозначения:

V_{taint} – множество всех переменных и блоков в адресных пространствах, «порожденных» в ходе анализа потока данных при обработке входных сообщений, элементы множества перезаписываемые.

I_{recv} – множество всех векторов, содержащих стек вызовов до вызова системной функции получения входных сообщений.

C_{proc} – множество, включающее заключенные в интервале между шагами получения и отправки сообщений, всех пар вида:

$$\langle addr_{call\_instr}, addr_{callee\_fn} \rangle,$$

где $addr_{call\_instr}$ – адрес инструкции вызова, $addr_{callee\_fn}$ – адрес вызываемой функции.

S_{num} – множество порядковых номеров выполнившихся инструкций.

$O_G = V_{taint} \times I_{recv} \times C_{proc} \times S_{num}$ – множество возможных наборов глобальных объектов протокола.

Описанный подход позволяет сформировать множество различающихся состояний изучаемого протокола, кроме того, можно утверждать, что существует некоторая функция $F: O_G \to S$, отображающая состояние множества глобальных объектов в множество состояний протокола. Функция F позволяет описать все возможные состояния. Очевидно, что функция F сюръективна, т.е. $\forall\, s \in S\, \exists\, o \in OG : F(o) = s$. Если предположить, что это не так, тогда в протоколе будет недостижимое состояние, другими словами, не являющееся состоянием протокола. Кроме того, это позволяет сделать предположения, что в процессе создания программного кода была допущена ошибка.

Множества P и Q – входных и выходных сообщений можно сформировать, осуществив перехват параметров функций приема и отправки сигналов.

Отметим связи между автоматами и некоторыми событиями потока управления, как например, выполнение функции `accept(…)` и `connect(…)` позволяет сделать вывод о выходе протокола из начального состояния и начале взаимодействия, `recv(…)` – получение входного сигнала, `send(…)` – потенциально, может служить индикатором перехода в следующее состояние, а `close(…)` – однозначно указывает на прекращение соединения и переход протокольного автомата в конечное состояние. Для построения функции переходов будем отталкиваться от описанных связей бинарного кода и математической модели. Кроме того, будем считать, что если один и тот же входной сигнал был успешно обработан более одного раза подряд и при этом были сформированы и отправлены одинаковые выходные сообщения, тогда протокольный автомат переходит в то же самое состояние, т. е. формируется петля в графе.

4. Описание разработанных макетов инструментов

Описанный в разд. 2 подход позволяет построить протокольный автомат, реализованный в бинарном коде, предполагающий штатное функционирование системы, однако наиболее интересной задачей представляется выявление дополнительных (сервисных) состояний, ошибок и недекларированных возможностей протокола, а также дефектов тестируемого бинарного кода.

Фаззинг протокола заключается в автоматической генерации входных сигналов, передаче их для обработки программе и последующей оценке состояния протокольного автомата.

В результате выполнения теста может либо осуществиться переход автомата из одного состояния в другое, либо нет, при этом если тест позволяет увеличить карту покрытия кода, обеспечивающего обработку входных данных, тогда этот тест также представляет интерес поскольку гарантирует получение сообщения отличного от предыдущих, что позволяет выявлять дефекты бинарного кода и уточнять грамматику для порождения входных сигналов.

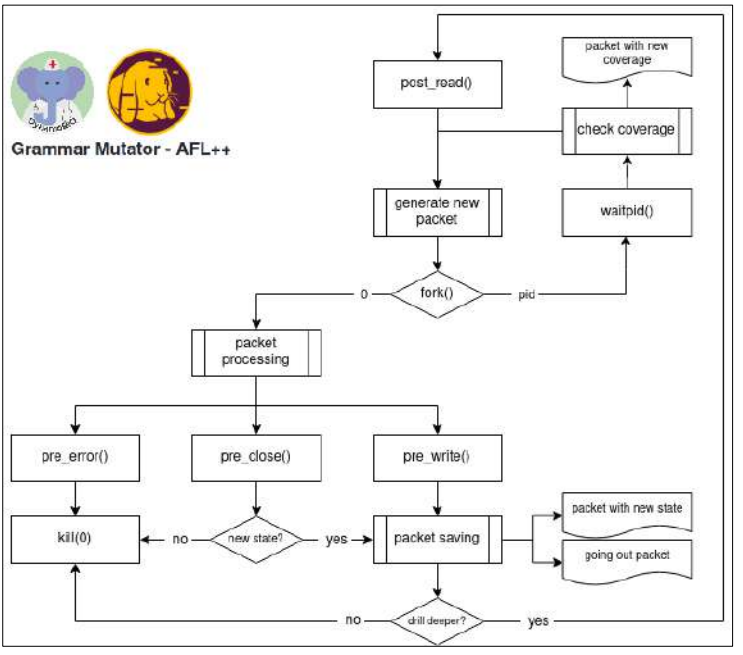


Рис. 7. Подход к восстановлению протокольного автомата сетевой программы (общая схема)
Fig. 7. Approach to recover protocol automata of a network application (common scheme)

В рамках реализации макета инструмента выделены три режима фаззинга протокола, в зависимости от решаемой задачи:

- фаззинг текущего состояний с целью увеличения покрытия;
- фаззинг с целью выполнения максимально возможного количества переходов автомата;
- комбинированный подход.

Общая схема реализованного подхода к фаззингу протокола представлена на рис. 7.

Суть метода заключается в создании динамической/плавающей точки запуска порождающего сервера (форк-сервера), обеспечивающего многократное повторения алгоритма на разных входных данных с учетом состояний протокола.

В качестве точки порождения выбирается инструкция вызова функции (системного вызова) получения входного сигнала, после чего генерируются новые данные, подменяются значения аргументов и возвращаемого значения функции (системного вызова) и анализируется процесс их обработки, включая граф потока управления и граф потока данных. Если тест заканчивается переходом на следующее состояние, тогда в зависимости от опций запуска либо осуществляется переход и выполняется порождение очередного потомка, либо оценивается покрытие, сохраняется результат и повторяется исходный тест.

Преимущество такого подхода – осуществление фаззинга непосредственно в памяти тестируемого процесса (in-memory fuzzing).

В данном контексте возникает три задачи:

- инструментация целевой программы для обеспечения возможности внедрения в ее код требуемых инструкций и выполнения «горячего» анализа потока данных и потока управления.
- генерация грамматически верных данных для уменьшения количества «пустых» тестов;
- оценка карты покрытия.

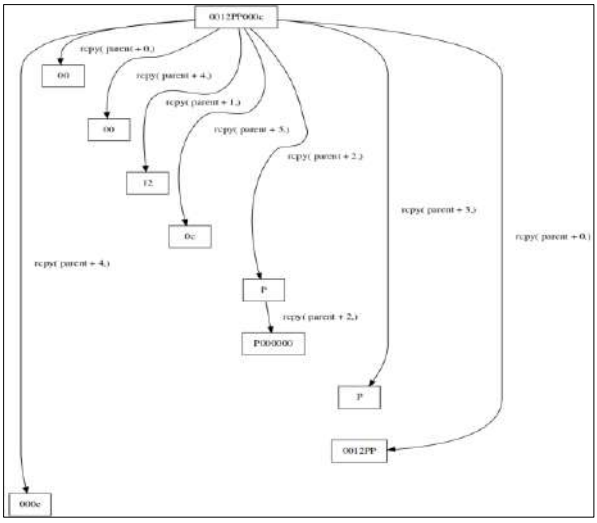


Рис. 8. Результат автоматического восстановления формата входного сообщения
Fig. 8. Automatic input signal format restoring result (graphical representation)

Для динамической инструментации при разработке инструментов выбрана система DynamoRIO – удобный инструмент для разработки инструментов анализа бинарного кода приложений, работающих в пользовательских режимах, DynamoRIO предназначена для функционирования в среде операционных систем (ОС) семейств Windows, Linux, Android,

экспериментальная версия поддерживает Mac. Допустимые архитектуры процессоров: IA-32, AMD64, ARM и AArch64.

Для изучения форматов входных сообщений разработан макет инструмента также с использованием системы DynamoRIO. Аналогично подходу, реализованному в инструменте Vigilante [27], алгоритм анализа потока данных, основан на инструментации инструкций передачи управления, чтения и записи данных адресных пространств, а также обертывании стандартных вызовов, используемых для работы с памятью, сокетами и функций криптографической библиотеки OpenSSL. Пример результата работы инструмента показан на рис. 8.

Генерация грамматически верных данных обеспечивается с помощью инструмента Grammar Mutator [23] из набора инструментов AFL++ [28]. В настоящее время грамматики описываются вручную на основе блоков входных сообщений, выделенных в процессе анализа их форматов.

Оценка покрытия выполняется методом, реализованным в фаззере AFL++.

Обобщенно реализованный подход описывается следующими шагами:

1) инструментируем целевую программу, запускаем, осуществляем взаимодействие с ответной частью системы, в результате формируется множество $P \times Q$, получаем описание форматов входных сообщений и протокольного автомата;
2) описываем грамматики входных сообщений, собираем библиотеки для генерации данных;
3) запускаем стенд, сохраняем максимально возможное количество пар из множества $P \times Q$, формируем начальную матрицу переходов, все непроверенные переходы заполняет нулями;
4) проводим эксперимент по отправке сообщений в произвольном порядке, уточняем матрицу;
5) запускаем процесс фаззинга протокола с использованием механизмов генерации данных на основе сформированных грамматик и учета покрытия кода, фиксируем новые пары из множества $P \times Q$, заполняем ранее нулевые поля, при необходимости расширяем матрицу, выявляем эквивалентные состояния, редуцируем автомат.

5. Описание экспериментов с VPN-клиентом

Цель экспериментов заключалась в оценке работоспособности предложенного метода на примере разработанных макетов инструментов.

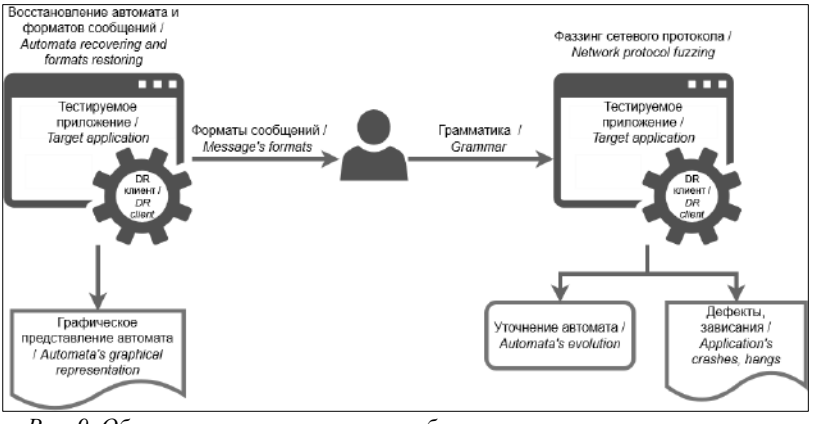


Рис. 9. Общая схема применения разработанных макетов инструментов
Fig. 9. Common scheme for using the developed tools

Задачи экспериментов:

- восстановление протокольного автомата, последовательностей и форматов сообщений;
- тестирование протокольного автомата методом фаззинга.

Для проведения верифицируемых экспериментов в качестве объекта был выбран VPN-клиент с открытыми исходными текстами, реализующий технологию защищенного SSL VPN туннеля. Общая схема применения разработанных макетов инструментов в рамках проведенных экспериментов показана на рис. 9.

На начальном этапе тестируемое приложение запускалось с целью восстановления форматов и последовательностей сообщений, а также протокольного автомата с помощью предназначенного для этого разработанного макета инструмента. В результате выделены три фазы сетевого функционирования клиента:

- процедура SSL-handshake, реализуемая с помощью библиотеки OpenSSL (анализ не проводился);
- авторизация на сервере, выполняемая с помощью протокола HTTP;
- функционирование канала на основе протокола PPP.

Для фаз подключения, в рамках которых осуществлялось взаимодействие с использованием протоколов HTTP и PPP, записаны последовательности входных и исходящих сообщений, а также в формате DOT созданы деревья разбора входных сообщений. Кроме того, получены данные для восстановления протокольного автомата. Графическое представление восстановленного автомата представлено на рис. 10, HTTP- и PPP-фазы отображены прямоугольниками с соответствующими текстовыми метками.

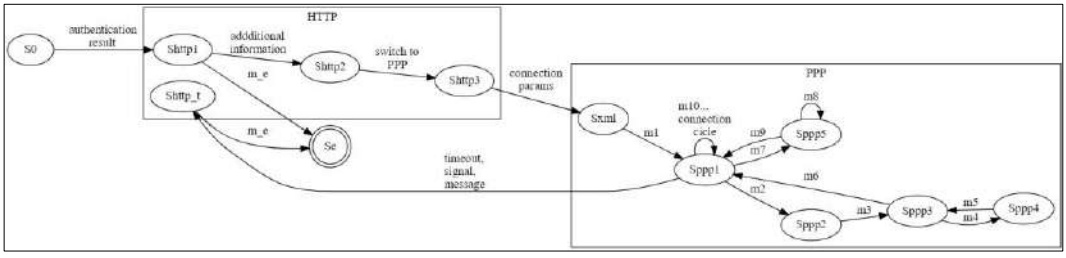


Рис. 10. Восстановленный протокольный автомат VPN-клиента (графическое представление)
Fig. 10. VPN-client's restored protocol automata (graphical representation)

На втором этапе экспериментов было выполнено изучение деревьев разбора входных сообщений (осуществлялось в ручном режиме), выделены ключевые поля, в результате разработана грамматика и инструмент, позволяющий генерировать псевдо HTTP-код (листинг 5).

```
{
 "<START_LINE>": [["HTTP/1.1 200 OK\r\n", "<HEADERS>", "\r\n\r\n",
                   "<BODY>", "\r\n\r\n"]],
 "<HEADERS>": [[], ["<HEADER>", "<DELIM>", "<HEADERS>"]],
 "<DELIM>": [[], ["\r"], ["\n"], ["\r\n"], ["\n\r"], ["\u0000"]],
 "<HEADER>": [["<HEADER_FIELD>", ": ", "<ANY>"]],
 "<HEADER_FIELD>": [["Content-Length"], ["Set-Cookie"]],
 "<URI>": [["<SCHEME>" , ":", "<HIER>", "<FRAGMENT>"]],
 "<SCHEME>": [["http"], ["https"], ["shttp"], ["dav"], ["attachment"],
              ["data"], ["file"], ["ftp"]],
 "<HIER>": [["//", "<AUTHORITY>", "<PATH>"]],
 "<AUTHORITY>": [["<USERINFO>", "<HOST>"]],
 "<PATH>": [["/", "<DIR>"]],
 "<DIR>": [[], ["<CHARS>", "/", "<DIR>"]],
 "<FRAGMENT>": [[], ["#", "<CHARS>"]],
 "<BODY>": [[], ["<CHARS>"]],
 "<SVPN>": [["SVPNCOOKIE="]],
```

```
  "<SVPNCOOKIE>": [["<SVPN>", "<CHARS>"]],
  "<ANY>": [[], ["<CHAR>"], ["<CHARS>"], ["<HOST>"], ["<URI>"],
            ["<SVPNCOOKIE>"]],
  "<CHARS>": [[],["<CHAR>", "<CHARS>", "<CHARS>", "<CHARS>"]],
  "<CHAR>": [["0"], ["1"], ["2"], ["3"], ["4"], ["5"], ["6"], ["7"],
             ["8"],
          ["9"], ["a"], ["b"], ["c"], ["d"], ["e"], ["f"], ["g"], ["h"],
          ["r"], ["s"], ["t"], ["u"], ["v"], ["w"], ["x"], ["y"], ["z"],
          ["="]]
}
```

Листинг 5. Грамматика для генерации входных данных в ходе фаззинга
Listing 5. Grammar for generating fuzzing input data

На третьем этапе экспериментов с использованием разработанного макета инструмента фаззинга протоколов, выполнено тестирование HTTP-фазы автомата. С использованием сформированной грамматики выполнялась генерация псевдо HTTP-кода, который использовался в качестве входных данных в процессе фаззинга.

В результате построена последовательность входных сообщений, позволяющая перейти из начального состояния *S0* к состоянию *Sppp1* (табл. 2). Кроме того, обнаружены сообщения, приводящие к зависанию VPN-клиента в состояниях *Shttp1*, *Shttp2*, *Shttp3*.

Табл. 2. Восстановленная последовательность входных сообщений
Table. 2. Recovered input sequence leading to Sppp1 state

Состояние	**Описание входного сообщения от сервера**	**Содержимое сгенерированного с помощью грамматики входного сообщения, позволяющего перейти к следующему состоянию**
Shttp1	HTTP сообщение размером 782 байта, содержащее заголовки и html-контент	HTTP/1.1 200 OK Set-Cookie: SVPNCOOKIE=
Shttp2	HTTP сообщение размером 2105 байт, содержащее заголовки и html-контент	HTTP/1.1 200 OK
Shttp3	HTTP сообщение размером 2099 байт, содержащее заголовки и html-контент	HTTP/1.1 200 OK
Sxml	HTTP сообщение размером 914 байт, содержащее заголовки и xml-контент	HTTP/1.1 200 OK Set-Cookie: ssh://<USERINFO><HOST>/ Content-Length: Content-Length: t
Sppp1	…	…

Анализ исходных текстов тестируемой программы показал, что зависания происходят из-за того, что разработчиками не предусмотрена возможность получения сообщения с некорректным содержимым. Предполагается, что все сообщения будут удовлетворять форматам данных, отправляемых сервером, в противном случае алгоритм считает, что получена только часть сообщения и переходит к ожиданию оставшихся данных и зависает на функции чтения из сети.

6. Преимущества подхода, ограничения и перспективы

Предлагаемый подход позволяет в короткое время в автоматизированном режиме построить протокольный автомат сетевого протокола, сгенерировать большой корпус данных, обеспечивающих переход автомата из состояния в состояние и позволяющих увеличивать покрытие кода, реализующего алгоритм протокола.

Анализ трассы выполнения программы позволяет заглянуть внутрь процесса инкапсуляции протоколов, проанализировать содержимое сигналов, которые зашифрованы или

закодированы при передаче по каналу связи, кроме того, в отличие от подходов основанных на анализе сетевых трасс, имеющих сложности при исследовании протоколов без обратной связи, данный подход позволяет в ходе анализа потока управления, оценивая результаты выполнения сетевых функций, однозначно идентифицировать некоторые состояния протокола.

На момент написания статьи на использование разработанных макетов накладываются ограничения в возможности исследования приложений, предназначенных для функционирования в ОС семейства Linux, на архитектуре процессора x86_64. Дополнительные ограничения в силу отсутствия достаточной статистики применения пока не определены.

В настоящее время выполняется решение следующих перспективных задач.

1) Разработка механизма для автоматического комбинированного фаззинга, позволяющего одновременно тестировать текущие состояния и выполнять переходы к следующим.
2) Разработка механизма использования ролевых моделей функций для обеспечения возможности управления алгоритмом инструментирования тестируемой программы без необходимости модификации исходных текстов.
3) Автоматизированное формирование скриптов с описанием начального протокольного автомата и формата входных сигналов, а также расширение способов генерации данных в ходе фаззинга протоколов.
4) Разработка способа формального описания протокольных автоматов.
5) Автоматизированное создание исполняемого файла ответной части исследуемой программы, обеспечивающей возможность воспроизведения сгенерированных последовательностей сообщений и выявленных ошибок реализации протокола, а также дефектов бинарного кода.
6) Создание версий, совместимой с архитектурой ARM в ОС Linux и предназначенной для использования в среде ОС Windows.
7) Расширение набора протестированных программ.

7. Заключение

В статье описан подход к восстановлению протокольных автоматов по бинарному коду сетевых программ и фаззингу протоколов, при этом разработаны:

- автоматизированный эвристический подход к определению типа протокола (о других существующих методах решения этой задачи автору ничего не известно), выявлению состояний и построению протокольного автомата в «первом приближении»;
- математическая модель протокольного автомата сетевой программы и метод ее проецирования на бинарный код;
- автоматизированный метод решения задач построения протокольного автомата и его уточнения с использованием бинарного кода «горячей» трассы выполнения программы;
- макеты инструментов для восстановления форматов сообщений сетевых протоколов и построения протокольного автомата в ходе анализа трассы выполнения программы и фаззинга сетевых протоколов, чувствительных к сохранению состояния, в памяти тестируемого процесса, с использованием «плавающей» точки запуска порождающего сервера.

Проведены тесты на примере реальной сетевой программы и представлены полученные результаты. В дальнейшем работы по данной тематике будут продолжены в соответствии с определенными перспективными задачами и направлениями.

Список литературы / References

[1] SMACC – State Machine Asynchronous C++. Available at: https://smacc.dev/behavioral-vs-protocol-state-machines/#, accessed 11.10.2021.

[2] Poll E. LangSec meets state machines. Presentation at the IT day at SWIFT, Belgium, 2017. Available at: https://www.cs.ru.nl/E.Poll/talks/SWIFT_2017.pdf, accessed 11.10.2021.

[3] Székely G., Ládi G. et al. Protocol State Machine Reverse Engineering with a Teaching-Learning. Acta Cybernetica, vol. 25, issue 2, 2021, pp. 517-535.

[4] Ládi G., Buttyán L., Holczer T. GrAMeFFSI: Graph analysis based message format and field semantics inference for binary protocols using recorded network traffic. Infocommunications Journal, vol. 12, issue 2, 2020, pp. 25-33.

[5] Shahbaz M., Groz R. Inferring mealy machines. Lecture Notes in Computer Science, vol. 5850, 2009, pp. 207-222.

[6] Sija B.D, Goo Y.-H. et al. A Survey of Automatic Protocol Reverse Engineering Approaches, Methods, and Tools on the Inputs and Outputs View. Security and Communication Networks, 2018, Article ID 8370341, 17 p.

[7] Shevertalov M., Mancoridis S.. A reverse engineering tool for extracting protocols of networked applications. In Proc. of the 14th Working Conference on Reverse Engineering (WCRE ’07), 2007, pp. 229-238.

[8] Xiao M.-M., Yu S.-Z., and Wang Y. Automatic network protocol automaton extraction. In Proc. of the 3rd International Conference on Network and System Security (NSS ’09), 2009, pp. 336-343.

[9] Trifilo A., Burschka S., Biersack E. Traffic to protocol reverse engineering. In Proc. of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1-8.

[10] Antunes J., Neves N. Building an automaton towards reverse protocol engineering. 2009. Available at: http://www.di.fc.ul.pt/~nuno/PAPERS/INFORUM09.pdf, accessed 11.10.2021.

[11] Antunes J., Neves N., Verissimo P. Reverse engineering of protocols from network traces. In Proc. of the 18th Working Conference on Reverse Engineering (WCRE ’11), 2011, pp. 169-178.

[12] Wang Y., Zhang Z. et al. Inferring protocol state machine from network traces: a probabilistic approach. In Proc. of the 9th Applied Cryptography and Network Security International Conference (ACNS ’11), 2011, pp. 1-18.

[13] Zhang Z., Wen Q.-Y., and Tang W. Mining protocol state machines by interactive grammar inference. In Proc. of the 2012 3rd International Conference on Digital Manufacturing and Automation (ICDMA ’12), 2012, pp. 524-527.

[14] Laroche P., Burrows A., and Zincir-Heywood A.N. How far an evolutionary approach can go for protocol state analysis and discovery. In Proc. of the IEEE Congress on Evolutionary Computation (CEC ’13), 2013, pp. 3228-3235.

[15] Meng F., Liu Y. et al. Inferring protocol state machine for binary communication protocol. In Proc. of the IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA ’14), 2014, pp. 870-874.

[16] Borisov N., Brumley D.J. et al. Generic application-level protocol analyzer and its language. MSR Technical Report MSR-TR-2005-133, 2005, 15 p.

[17] Wang Y., Li X. et al. Biprominer: automatic mining of binary protocol features. In Proc. of the 12th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT ’11), 2011, pp. 179-184.

[18] Bossert G., Guihéry F., and Hiet G. Towards automated protocol reverse engineering using semantic information. In Proc. of the 9th ACM Symposium on Information, Computer and Communications Security, 2014. Pp. 51-62.

[19] Bossert G., Guihéry F. Reverse and simulate your enemy botnet C&C. Mapping a P2P Botnet with Netzob. In Proc. of the Black Hat Abu Dhabi Conference, 2012, 21 p.

[20] Luo J.-Z. and Yu S.-Z. Position-based automatic reverse engineering of network protocols. Journal of Network and Computer Applications, vol. 36, issue 3, 2013, pp. 1070-1077.

[21] Comparetti P.M., Wondracek G. et al. Prospex: protocol specification extraction. In Proc. of the 30th IEEE Symposium on Security and Privacy, 2009, pp. 110-125.

[22] Xiao M.-M. and Luo Y.-P.. Automatic protocol reverse engineering using grammatical inference. IFS, vol. 32, no. 5, pp. 3585–3594, Apr. 2017, DOI: 10.3233/JIFS-169294.

[23] Goo Y.-H., Shim K.-S. et al. Protocol Specification Extraction Based on Contiguous Sequential Pattern Algorithm. IEEE Access, vol. 7, 2019, pp. 36057-36074.

[24] List of (automatic) protocol reverse engineering tools/methods/approaches for network protocols. Available at: https://github.com/techge/PRE-list, accessed 14.10.2021.
[25] Costa M., Crowcroft J. et al. Vigilante: End-To-End Containment of Internet Worms. In Proc. of the 20th ACM Symposium on Operating Systems Principles (SOSP 2005), 2005, pp. 133-147.
[26] Lang K.J. Faster Algorithms for Finding Minimal Consistent DFAs. Technical Report. NEC Research Institute, 1999, 19 p.
[27] Needleman S. and Wunsch C. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. Journal of Molecular Biology, vol. 48, issue 3, 1970, pp. 443-453.
[28] Grammar-Mutator. Available at: https://github.com/AFLplusplus/Grammar-Mutator, accessed 22.09.2022.
[29] AFLplusplus. Available at: https://github.com/AFLplusplus/AFLplusplus.git, accessed 20.09.2022.

Информация об авторах / Information about authors

Иван Владимирович ШАРКОВ – научный сотрудник отдела компиляторных технологий. Сфера научных интересов: информационные технологии, системное программирование, фаззинг, компьютерные сети, исследования программ.

Ivan Vladimirovich SHARKOV is a researcher at the compiler technologies department. His research interests include information technologies, fuzzing, networks, software researching.

DOI: 10.15514/ISPRAS-2022-34(5)-4

Способ оценки похожести программ методами машинного обучения

[1] *П.Д. Борисов, ORCID: 0000-0002-8919-8310 <borisovpetr@mail.ru>*
[2] *Ю.В. Косолапов, ORCID: 0000-0002-1491-524X <itaim@mail.ru>*

[1] *ФГАНУ НИИ «Спецвузавтоматика»*
344003, Россия, г. Ростов-на-Дону, ул. Города Волос, 6
[2] *Южный федеральный университет,*
344006, Россия, г. Ростов-на-Дону, ул. Б. Садовая, д. 105/42

Аннотация. В работе рассматривается задача построения алгоритма сравнения двух исполнимых файлов. В основе алгоритма лежит построение по заданной паре программ вектора показателей похожести и принятие на основе этого вектора решения о похожести или непохожести программ с помощью методов машинного обучения. Показатели похожести строятся с помощью алгоритмов двух типов: с помощью алгоритмов, не учитывающих формат входных данных (значения нечетких хеш-функций, значения коэффициентов сжатия), и с помощью алгоритмов, выполняющих анализ машинного кода (с помощью дизассемблеров). Всего построено 15 показателей: 9 показателей первого типа и 6 второго. На основе построенного обучающего множества пар похожих и непохожих программ (на основе набора программ coreutils) обучены и протестированы 7 разных бинарных классификаторов. Результаты экспериментов показали высокую точность моделей на основе случайного леса и k ближайших соседей. Также выявлено, что совместное применение показателей обоих типов может увеличить точность классификации.

Ключевые слова: обфускация; похожесть программ; машинное обучение

Для цитирования: Борисов П.Д., Косолапов Ю.В. Способ оценки похожести программного кода методами машинного обучения. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 63-76. DOI: 10.15514/ISPRAS-2022-34(5)-4

A Method to Evaluate Program Similarity Using Machine Learning Methods

[1] *P.D. Borisov, ORCID: 0000-0002-8919-8310 <borisovpetr@mail.ru>*
[2] *Yu.V. Kosolapov, ORCID: 0000-0002-1491-524X <itaim@mail.ru>*

[1] *FSASE SRI «Specvuzavtomatika»*
6, st. Goroda Volos, Rostov-on-Don, 344003, Russia
[2] *Southern Federal University,*
105/42, Bol'shaya Sadovaya st., Rostov-on-Don, 344006, Russia

Abstract. The problem of constructing an algorithm for comparing two executable files is considered. The algorithm is based on the construction of similarity features vector for a given pair of programs. This vector is then used to decide on the similarity or dissimilarity of programs using machine learning methods. Similarity features are built using algorithms of two types: universal and specialized. Universal algorithms do not take into account the format of the input data (values of fuzzy hash functions, values of compression ratios). Specialized algorithms work with executable files and analyze machine code (using disassemblers). A total of 15 features were built: 9 features of the first type and 6 of the second. Based on the constructed training set of similar and dissimilar program pairs, 7 different binary classifiers were trained and tested. To build the training

set, coreutils programs were used. The results of the experiments showed high accuracy of models based on random forest and k nearest neighbors. It was also found that the combined use of features of both types can improve the accuracy of classification.

Keywords: obfuscation; program similarity; machine learning

For citation: Borisov P.D., Kosolapov Yu.V. A method for evaluating the similarity of program code using machine learning methods. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 63-76 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-4

1. Введение и постановка задачи

Преобразования программного кода, сохраняющие его функциональность, можно разделить на оптимизирующие, диверсифицирующие и обфусцирующие [1]. Первые применяются для ускорения работы программы и/или уменьшения размера программы, а вторые – для защиты программ от удаленных атак, так как эксплойт, предназначенный для оригинальной версии программы, может не подходить для диверсифицированного варианта [2]. Целью же обфусцирующих преобразований обычно является сокрытие (маскировка) алгоритмов, используемых в программе, и/или сокрытие критически важных данных, например, криптографических ключей [3]. Основным отличием обфусцирующих преобразований от оптимизирующих и диверсифицирующих является их направленность на затруднение понимания программы аналитиком как при статическом, так и при динамическом анализе. Обычно в результате таких преобразований обфусцированная версия *Obf(P)* программы *P* становится, нестрого выражаясь, «менее похожа» на необфусцированный вариант *P*. Отметим, что термин «похожесть программ» не является четко определенным в программной инженерии, что является следствием неточности самого слова «похожесть». Тем не менее, исследователи сходятся в возможности определения похожести на синтаксическом (структурном) и/или на семантическом (поведенческом) уровне [4]. Именно непохожесть *Obf(P)* и *P* на синтаксическом или семантическом уровнях, например, позволяет маскировать известный код под неизвестный код, а также затрудняет восстановление проприетарного алгоритма или секретных данных из кода. В связи с этим, конкурирующими направлениями исследований являются разработка эффективных методов обнаружения похожести программ и разработка практических методов обфускации, преобразующих программу *P* в непохожую на нее версию *Obf(P)*.

В настоящей работе похожесть программ определяется на семантическом уровне, причем под программой понимается исполнимый файл программы, полученный по исходному коду с помощью компилятора. Далее в тексте две программы считаются похожими, если на одинаковых входных данных возвращают одинаковый результат, и при этом они реализуют один и тот же алгоритм. В рамках такого определения похожести две программы, например, реализующие разные алгоритмы сортировки и возвращающие на одинаковых входных данных одинаковый результат, считаются непохожими. И хотя такое определение существенно сужает класс пар похожих программ, оно представляется удобным в рамках следующей модели оценки стойкости обфусцирующих преобразований, предложенной в [5]. Пусть $Similarity$ – метод вычисления похожести двух программ, возвращающий действительное число из диапазона [0,1], где 0 – программы похожи, 1 – непохожи. Для программы P преобразование Obf_1 считается более стойким, чем Obf_2, если

$$Similarity(P, Obf_1(P)) > Similarity(P, Obf_2(P)).$$

В частном случае, когда $Similarity$ возвращает «жесткое» решение: 0 либо 1 (похожи, либо непохожи), преобразование Obf считается стойким, если $Similarity\big(P, Obf(P)\big) = 1$, и нестойким в противном случае. Здесь и далее предполагается, что метод $Similarity$ сравнения программ строиться без учета известных алгоритмов обфусцирующих преобразований, так как при таком подходе представляется возможным объективное сравнение стойкости разных методов обфускации. В этом случае метод $Similarity$, как с

«мягкими» решениями, так и с «жесткими» решениями, может применяться для итеративного подбора обфусцирующего преобразования (из заданного множества преобразований), обеспечивающего требуемый уровень стойкости. Целью подбора является выбор такого преобразования Obf для программы P, при котором метод $Similarity$ будет считать программы $Obf(P)$ и P наименее похожими – для алгоритма с «мягкими» решениями, и непохожими – для алгоритма с «жесткими» решениями. Отметим, что аналогичный подход к оценке стойкости обфусцирующих преобразований реализуется в [6], где стойкость оценивается не с помощью похожести файлов, а с помощью времени символьного исполнения программ.

В методе $Similarity$ при сравнении P и Q могут учитываться как статические характеристики, получаемые без запуска программ, так и динамические, вычисляемые в результате запуска или симуляции запуска. В [7,8] исследуется возможность применения в $Similarity$ в качестве динамических характеристик таких, которые могут быть построены с помощью разработанной в [5] схемы получения характеристик символьного исполнения. Настоящее исследование направлено на применение в $Similarity$ только статических характеристик. Стоит отметить, что совместное применение как статических, так и динамических характеристик, может позволить повысить точность $Similarity$, однако эта задача является направлением дальнейшего исследования и в этой работе не решается.

Способы статической оценки степени похожести условно можно разделить на два типа: универсальные и специализированные. К универсальным отнесем способы, не учитывающие то, что сравниваемыми объектами являются файлы/фрагменты исполнимого кода. В то время как способы, учитывающие этот аспект, отнесем к специализированным. В работе с помощью методов машинного обучения строится и исследуется алгоритм, который по двум исполнимым файлам P и Q на основании вычисленного значения из диапазона действительных чисел [0,1] возвращает «жесткое» решение: 0 – программы похожи, 1 – программы непохожи. Вектор признаков строиться на основе значений, вычисленных с помощью универсальных и специализированных способов сравнения программ P и Q. Целью исследования является оценка точности некоторых известных моделей классификации и оценка эффекта от совместного применения универсальных и специализированных подходов сравнения.

2. Существующие подходы к оценке похожести

Как отмечалось выше, способы статической оценки степени похожести исполнимых файлов можно разделить на универсальные и специализированные. К универсальным способам относятся способы на основе вычисления значений нечетких хеш-функций (fuzzy hash), которые, в отличие от криптографических хеш-функций, слабо чувствительны к малым изменениям аргумента. Обычно алгоритмы нечетких хеш-функций не учитывают формат данных: используется один алгоритм вычисления значения для машинного кода, текстовых данных, аудиоданных и видеоданных, в чем и заключается их универсальность. Такие функции, например, успешно применяются в системах фильтрации электронных писем, при криминалистическом анализе цифровых документов, при выявлении версий вредоносных файлов (в таких сервисах, как VirusTotal, VirusShare).

Однако результаты исследования [9] показали, что не все нечеткие хеш-функции одинаково эффективны при оценке похожести бинарных файлов программ. В частности, хеш-функция ssdeep реже других, рассмотренных в [9] функций, обнаруживает похожие программы. Там же отмечено, что обфускация может существенно менять бинарный код, что может снизить качество обнаружения похожих программ для любой хеш-функции (не только ssdeep). Это связано с тем, что нечеткие хеш-функции устойчивы только к малым изменениям. Вопрос совместного применения нескольких хеш-функций для повышения качества обнаружения похожих программ в [9] не исследовался.

В универсальности способов на основе нечетких хеш-функций заключается и низкое качество результатов сравнения похожих исполнимых файлов. За счет учета формата данных может повыситься качество сравнения похожих программ, но при этом универсальность хеш-функции будет утрачена. В [10, 11] фактически предлагаются специализированные подходы построения нечетких хеш-функций для исполнимых файлов. Именно, в [10, 11] решается следующая задача: для заданного объекта сравнения (функции/подпрограммы) найти наиболее близкий объект в заданном списке (*архивном множестве* объектов). Для решения этой задачи в [10, 11] на основе дизассемблированного кода строится описание объекта сравнения. Это описание, представляющее собой вектор фиксированной длины, можно рассматривать как значение нечеткой хеш-функций от объекта сравнения, представляющего собой машинный код. В [10] для построения описания используется нейросеть, а в [11] вектор описания объекта сравнения предлагается строить как набор значений мер схожести с каждым элементом *опорного множества* объектов (используются пять мер схожести). Для вычисления меры похожести двух векторов описаний в [10] применяется косинусный коэффициент, а в [11] – обобщенный коэффициент Жаккара. В обоих случаях для применения методов сравнения необходимо иметь архивное множество объектов, с помощью которого может быть построено описание объектов сравнения.

Другие специализированные подходы к оценке похожести программ предлагаются в [12] и [13], где решается задача поиска клонов исходного кода на основе семантического анализа программ. Именно, в [12] и [13] для повышения точности обнаружения клонов предлагается сравнивать графы зависимостей программ. Недостатком такого подхода, как отмечается в [12], является его относительно низкая скорость, хотя в [13] и удалось кроме повышения точности сравнения добиться достаточно высокой скорости сравнения. Заметим, что в свободном доступе не удалось найти программных реализаций подходов из [10,11], что затрудняет их сравнение как между собой, так и с другими подходами. Способы из [12] и [13] хотя и реализованы авторами, также не находятся в свободном доступе.

Кроме того, представляется, что эти подходы в меньшей степени (чем подходы из [10,11]) соответствуют задаче оценки стойкости обфусцирующего преобразования по следующим причинам. Во-первых, методы поиска клонов, в частности из [12] и [13], нацелены на выявление трех типов клонов (см. [14]), которые возникают именно в контексте клонирования кода, когда в первую очередь перед разработчиком стоит задача адаптации готового кода под собственные нужды, а задача запутывания кода при этом не является первоочередной. Во-вторых, важными характеристиками алгоритмов поиска клонов являются точность и скорость работы, в то время как при оценке стойкости обфусцирующих преобразований наибольшую роль играет точность, при этом скорость существенна в меньшей степени. В-третьих, для выявления клонов кода алгоритму поиска могут потребоваться исходные тексты программ, а при выявлении похожести исполнимых файлов сравниваются скомпилированные версии программ, что позволяет оценивать любые обфусцируюище преобразования и абстрагироваться от уровня их применения (уровня исходного кода, уровня промежуточного представления и уровня машинного кода).

3. Способ оценки похожести исполнимых файлов

2.1 Начальное описание программ

В настоящем исследовании похожесть программ оценивается на основе ряда статических характеристик с привлечением методов машинного обучения. Для каждой программы *P* ее *начальное описание* задается как множество статических характеристик двух типов: множество универсальных характеристик состоит из тех, которые получены с помощью средств, не учитывающих формат данных (значения нечетких хеш-функций, значения коэффициентов сжатия); во множество специализированных характеристик отнесены те,

которые получены с помощью средств анализа машинного кода (в частности, с помощью дизассемблеров). Саму программу *P* можно отнести как к универсальным характеристикам, так и к специализированным характеристикам. Это связано с тем, что многие утилиты сравнения файлов, в качестве входных данных принимают сами файлы. При этом одни утилиты рассматривают файлы как набор бинарных данных (утилиты с универсальным способом сравнения), а другие – как исполнимый код (утилиты со специализированным способом сравнения).

Использование значений нечетких хеш-функций для начального описания обосновано простотой их использования и тем, что они уже применяются для сравнения файлов с исполнимым кодом. В частности, эти функции находят применение в криминалистическом анализе вредоносного кода [15], в онлайн-сервисах обнаружения вредоносного кода, таких как VirusTotal [16]. Применение алгоритмов сжатия обосновано, с одной стороны простотой их применения, а с другой стороны, тем, что они могут использоваться как аппроксимация нормализованного расстояния сжатия [4]. В [17], например, Колмогоровская сложность программы используется как оценка качества обфусцирующих преобразований, а для аппроксимации этой сложности применяются алгоритмы сжатия. Использование коэффициентов сжатия в начальном описании программы, таким образом, основано на предположении, что похожие по функциональности необфусцированные программы, имеют близкую сложность по Колмогорову.

Для получения специализированных характеристик в настоящей работе применяются дизассемблеры, позволяющие по файлу с исполнимым кодом построить его представление в формате BinExport [18], пригодном для сравнения с помощью утилиты BinDiff [19]. Формат BinExport позволяет хранить граф потока управления и граф вызова функций, которые используются утилитой BinDiff для вычисления расстояния между исполнимыми файлами. Выбор BinDiff обоснован, с одной стороны, тем, что, как показывают результаты исследования [20], подход, используемый в BinDiff, дает высокую точность сравнения; с другой стороны, в настоящий момент формат BinExport поддерживается популярными дизассемблерами IDA Pro [21], Ghydra [22] и Binary Ninja [23]. В частности, соответствующие плагины для IDA Pro и Ghydra позволяют автоматизировать процесс выгрузки данных в формате BinExport. Утилита Binary Ninja таких средств не предлагает, однако ее прикладной программный интерфейс позволяет автоматизировать этот процесс на языке Python.

Таким образом, для каждой программы *P* в базе данных сохраняется ее начальное описание, включающее характеристики, перечисленные в табл. 1.

Табл. 1. Начальное описание программы P, 0 – универсальная характеристика, 1 – специализированная характеристика

Table 1. Initial description of program P, 0 - universal characteristic, 1 - specialized characteristic

Характеристика		**Тип значения**	**Описание**
Обозначение	**Тип**		
P	0/1	Строка	Путь к файлу программы *P*
ssdeepHash	0	Строка	Значение нечеткой хеш-функции ssdeep (см. [24])
sdhashHash		Строка	Значение нечеткой хеш-функции sdhash (см. [25])
tlshHash		Строка	Значение нечеткой хеш-функции tlsh (см. [25])
lzmaVal		Число (float)	Степень сжатия программы *P* алгоритмом lzma (см. [26])
bz2Val		Число (float)	Степень сжатия программы *P* алгоритмом bz2 (см. [27])
deflateVal		Число (float)	Степень сжатия программы *P* алгоритмом deflate (см. [28])

idaBinExp	1	Строка	Путь к каталогу, содержащему представление программы в формате BinExport, полученное с помощью IDA Pro
ghydraBinExp		Строка	Путь к каталогу, содержащему представление программы в формате BinExport, полученное с помощью Ghydra
ninjaBinExp		Строка	Путь к каталогу, содержащему представление программы в формате BinExport, полученное с помощью Binary Ninja

2.2 Описание похожести программ

Для пары программ *P* и *Q* по их начальным описаниям строится вектор *sim*(*P*,*Q*) из пятнадцати показателей похожести, каждый из которых принимает действительное значение от нуля до единицы: значение 0 соответствует похожим программам, а значение 1 – непохожим. Вектор *sim*(*P*,*Q*) включает показатели из табл. 2, где приведены условные наименования показателей и способ их вычисления. Показатели, вычисленные по универсальным (соответственно специализированным) характеристикам, в табл. 2 называются универсальными (соответственно специализированными) показателями. На основании вектора *sim*(*P*,*Q*) в дальнейшем с помощью методов машинного обучения будет приниматься решение о похожести или непохожести *P* и *Q*.

Отметим, что утилита BinDiff, сравнивающая графы потока управления программ *P* и *Q*, возвращает два значения: значение похожести программ и степень уверенности (confidence). Поэтому в число показателей в табл. 2 для каждого из трех рассмотренных дизассемблеров, поддерживающих формат BinExport, входят как значение похожести, возвращенное BinDiff (строки 10, 12, 14 в табл. 2), так и произведение значения похожести на степень уверенности (строки 11, 13, 15 в табл. 2). Пакет анализа исполнимого кода Radare2 [29] не поддерживает формат BinExport, что не позволяет применить утилиту BinDiff. Однако Radare2 содержит утилиту radiff2, позволяющую находить расстояние Майерса [30] и расстояние Левенштейна [31]. Так как при сравнении дизассемблирование не выполняется, то показатели *myersdiff* (строка 8 в табл. 2) и *levenshteindiff* (строка 9 в табл. 2) являются универсальными.

Табл. 2. Показатели похожести программ P и Q, тип 0 – универсальный показатель, тип 1 – специализированный показатель

Table 2. Program similarity features P and Q, type 0 - universal feature, type 1 - specialized feature

№	Показатель	Тип	Описание
1	*ssdeep*	0	Результат сравнения *ssdeepHash(P)* и *ssdeepHash(Q)* утилитой сравнения хеш-значений функции ssdeep
2	*sdhash*		Результат сравнения *sdhashHash(P)* и *sdhashHash(Q)* утилитой сравнения хеш-значений функции sdhash
3	*tlsh*		Результат сравнения *tlshHash(P)* и *tlshHash(Q)* утилитой сравнения хеш-значений функции *tlsh* с учетом длины файлов *P* и *Q*
4	*tlshxlen*		Результат сравнения *tlshlenHash(P)* и *tlshlenHash(Q)* утилитой сравнения хеш-значений функции *tlsh* без учета длины файлов *P* и *Q*
5	*lzma*		$1 - \frac{\lvert lzmaVal(P) - lzmaVal(Q)\rvert}{\max\{lzmaVal(P), lzmaVal(Q)\}}$
6	*bz2*		$1 - \frac{\lvert bz2Val(P) - bz2Val(Q)\rvert}{\max\{bz2Val(P), bz2Val(Q)\}}$
7	*deflate*		$1 - \frac{\lvert deflateVal(P) - deflateVal(Q)\rvert}{\max\{deflateVal(P), deflateVal(Q)\}}$
8	*myersdiff*		Результат сравнения бинарных файлов *P* и *Q* с помощью утилиты radiff с опцией -s (расстояние Майерса) из пакета анализа программ Radare2
9	*levenshteindiff*		Результат сравнения бинарных файлов *P* и *Q* с помощью утилиты radiff с опцией -ss (расстояние Левенштейна)

10	*idadiff*	1	Результат сравнения представлений *idaBinExp(P)* и *idaBinExp(Q)* с помощью утилиты BinDiff (см. [19])
11	*idadiffc*		$idadiff \cdot confidence_{ida}$
12	*ghydradiff*		Результат сравнения представлений *ghydraBinExp(P)* и *ghydraBinExp(Q)* с помощью утилиты BinDiff
13	*ghydradiffc*		$ghydradiff \cdot confidence_{ghydra}$
14	*ninjadiff*		Результат сравнения представлений *ninjaBinExp(P)* и *ninjaBinExp(Q)* с помощью утилиты BinDiff
15	*ninjadiffc*		$ninjadiff \cdot confidence_{ninja}$

2.3 Алгоритм сравнения программ

Алгоритм сравнения программ *P* и *Q* предлагается строить следующим образом: 1) на основе входных файлов *P* и *Q* строится вектор показателей похожести *sim*(*P*,*Q*); 2) вектор *sim*(*P*,*Q*) передается обученному бинарному классификатору, который принимает решение о похожести (значение 0) или непохожести программ (значение 1). Стоит отметить, что «жесткое» решение – 0 или 1 – классификатором обычно принимается на основе «мягкого» решения из диапазона [0,1] и порога: при превышении порога «мягкое» решение преобразуется в «жесткое» решение 1, а при недостижении порога возвращается «жесткое» решение 0. Для реализации алгоритма сравнения программ необходимо решить две задачи: построить обучающее множество и выбрать лучший классификатор.

В то время, как выбор наилучшего классификатора может быть осуществлен путем сравнения результатов экспериментов, построение обучающего множества является отдельной задачей, в рамках которой следует определить достаточный для обучения объем и состав обучающей выборки. Обучающее множество должно состоять из двух непересекающихся подмножеств: подмножества пар похожих программ и подмножества пар разных, непохожих программ. Каждая пара похожих программ *P* и *Q* используется для вычисления соответствующего вектора *sim*(*P*,*Q*); аналогично вектор *sim*(*P*,*Q*) вычисляется и для каждой пары непохожих программ *P* и *Q*. Таким образом, по множествам похожих и непохожих пар программ могут быть построены соответствующие множества векторов показателей похожести, которые далее используются для обучения классификаторов.

Отметим, что, как множество похожих, так и множество непохожих пар программ, как представляется, должно строиться на основе программ, реализующих алгоритмы из разных предметных областей: программы обработки строк, утилиты по работе с файловой системой, криптографические утилиты, алгоритмы обработки сигналов, алгоритмы линейной алгебры и т.п. Такой подход к построению обучающего множества может позволить достичь универсальности метода $Similarity$ выявления похожести программ, без привязки к предметной области алгоритмов, реализуемых сравниваемыми программами.

В настоящей работе для построения обучающего множества предлагается использовать известный набор разных (по предметным областям реализуемых алгоритмов) программ с исходным кодом. Наличие исходного кода позволяет, используя разные компиляторы и опции компиляции (например, опции оптимизации) строить пары похожих программ: любая пара программ, полученных из одного исходного кода, но с помощью разных компиляторов и/или разных опций компиляций, считается парой похожих программ (в смысле определения похожести, данного во введении). При этом любая пара программ, полученных из разных исходных кодов, независимо от того, какие компиляторы и опции компиляции использовались, образует пару непохожих программ.

4. Результаты экспериментов

В работе для составления обучающего множества был выбран набор программ coreutils [32], включающий 106 программ, написанных на языке Си. Для компиляции использовалось 9 вариантов компиляторов (GCC [33] версий 7.5.0, 8.4.0, 9.4.0, 10.3.0, Clang [34] версий 7.0.1, 8.0.1, 9.0.1, 10.0.0, AOCC [35] версии 3.0.0) для операционной системы Linux, в каждом из которых применялись 5 опций оптимизации (O0, O1, O2, O3, Os). Таким образом, для каждой программы P создается 45 ее вариантов: $P_1, \dots, P_{45}$, – а всего для набора coreutils создается $45 \cdot 106 = 4770$ программ. Для каждой программы P можно построить $C_{45}^2 = 990$ пар похожих программ: $\{(P_i, P_j): i, j = 1, \dots, 45, i \neq j\}$. Поэтому множество векторов показателей похожести, соответствующих всем парам похожих программ, имеет мощность $990 \cdot 106 = 104940$.

Множество пар непохожих программ состоит из $C_{106}^2 \cdot 45^2 = 11269125$ пар, так как каждая программа в паре непохожих программ может быть выбрана 45-ю способами, причем по набору coreutils из 106 программ можно построить C_{106}^2 пар. Таким образом, число непохожих пар почти в 108 раз больше множества пар похожих программ. Для уменьшения дисбаланса в соотношении похожих/непохожих пар программ множество векторов показателей непохожих программ строится на основе пар, полученных в рамках фиксированного компилятора и фиксированной опции оптимизации.

Таким образом, можно построить $C_{106}^2 \cdot 45 = 250425$ векторов показателей. Для построения обучающих и тестовых выборок из числа непохожих случайным образом выбираются 104940 пары, а из числа похожих пар используются все 104940. Построение обучающих и тестовых выборок, а также обучение выполнено с помощью пакета sklearn [36] для языка Python. Для экспериментального выбора лучшего классификатора исследуются классификаторы *GaussianNB* (*GNB*), *LogisticRegression* (*LR*), *SVM KNeighborsClassifier* (*KNC*), *LinearDiscriminantAnalysis* (*LDA*), *RandomForestClassifier* (*RFC*), *DecisionTreeClassifier* (*DTC*) из пакета sklearn.

Табл. 3. Результаты обучения классификаторов
Table 3. Learning outcomes of classifiers

Класси-фикатор	Номер эксп-та	Precision (Похожие)	Precision (Непохожие)	Recall (Похожие)	Recall (Непохожие)	Accuracy
SVM	1	0,89	0,87	0,87	0,90	0,88
	2	0,99	0,82	0,78	0,99	0,89
	3	0,98	0,90	0,89	0,98	0,93
DTC	1	0,91	0,91	0,91	0,91	0,91
	2	0,94	0,93	0,93	0,94	0,93
	3	0,97	0,97	0,97	0,97	0,97
RFC	1	0,95	0,94	0,94	0,95	0,94
	2	0,98	0,93	0,93	0,98	0,95
	3	**0,99**	**0,98**	**0,98**	**0,99**	**0,98**
LDA	1	0,90	0,85	0,84	0,90	0,87
	2	0,98	0,82	0,78	0,98	0,88
	3	0,96	0,87	0,86	0,96	0,91
KNC	1	0,92	0,94	0,94	0,92	0,93
	2	0,96	0,92	0,92	0,96	0,94
	3	**0,99**	**0,98**	**0,98**	**0,99**	**0,98**
LR	1	0,88	0,86	0,86	0,88	0,87
	2	0,95	0,83	0,81	0,96	0,89
	3	0,94	0,89	0,88	0,94	0,91
GNB	1	0,71	0,62	0,53	0,78	0,65
	2	0,66	0,75	0,81	0,58	0,70
	3	0,68	0,73	0,76	0,63	0,7

В работе проведено пять экспериментов: эксперимент 1 – обучение классификаторов только на множестве универсальных показателей, эксперимент 2 – обучение только на множестве специализированных показателей, эксперимент 3 – обучение на всех показателях, эксперимент 4 – обучение классификаторов только на множестве универсальных показателей с исключением ряда показателей, эксперимент 5 – обучение на множестве всех показателей с исключением ряда универсальных. Целью первых трех экспериментов является исследование эффекта от совместного использования универсальных и специализированных показателей, а цель четвертого и пятого – исследовать вклад отдельных универсальных показателей в точность обучения классификаторов.

Результаты первых трех экспериментов приведены в табл. 3, где каждому классификатору соответствуют три строки значений: значения *i*-ой строки получены в рамках эксперимента *i*, *i*=1,2,3. Из табл. 3 видно, что классификаторы на основе специализированных показателей (вторая строка для каждого классификатора) обладают ожидаемо большей точностью (accuracy), чем классификаторы на основе универсальных показателей (первая строка). Тем не менее, совместное применение этих показателей увеличивает точность для всех рассмотренных классификаторов (третья строка каждого классификатора). Это свидетельствует о целесообразности совместного применения всех показателей из табл. 2. Из табл. 3 также видно, что классификаторы *RFC* и *KNC* демонстрируют наибольшую точность при сравнении похожих и непохожих программ; наименьшую точность продемонстрировал классификатор *GNB*.

С целью оценки влияния отдельных показателей на решение классификатора, штатными средствами пакета sklearn для классификаторов *RFC* и *DTC* получены веса показателей во всех первых трех экспериментах (см. табл. 4). Анализ весов показателей в табл. 4 для эксперимента 3 показывает, что средний вес специализированных показателей больше среднего веса универсальных показателей. В частности, для *DTC* суммарный вес первых равен 0.711, а для *RFC* – 0.632. Откуда для *DTC* средний вес специализированных показателей равен 0.118, а средний вес универсальных характеристик – 0.032; для *RFC* средний вес специализированных характеристик равен 0.105, а средний вес универсальных характеристик – 0.041. Таким образом, показатели на основе специализированных характеристик вносят больший вклад в принятие решения о похожести программ. При этом стоит отметить, что среди показателей на основе таких характеристик наибольшим весом, как в случае *DTC*, так и в случае *RFC*, обладает ninjadiff. Это может быть связано с тем, что перед выгрузкой данных в формате BinExport, исполнимый файл анализировался трижды (используя прикладной программный интерфейс пакета Binary Ninja), в то время как для дизассемблеров IDA Pro и Ghydra, анализ выполнялся только один раз.

Табл. 4. Веса показателей для классификаторов DecisionTreeClassifier и RandomForestClassifier
Table 4. Feature weights for the DecisionTreeClassifier and RandomForestClassifier classifiers

Показатель	**Тип показателя**	**Эксперимент 1**		**Эксперимент 2**		**Эксперимент 3**	
		DTC	*RFC*	*DTC*	*RFC*	*DTC*	*RFC*
ssdeep	0	**0,019**	**0,016**	-	-	**0,003**	**0,004**
sdhash		0,117	0,140	-	-	0,086	0,079
tlsh		0,181	0,151	-	-	**0,012**	0,036
tlshxlen		0,300	0,256	-	-	0,025	0,078
lzma		**0,046**	**0,061**	-	-	**0,006**	**0,015**
bz2		**0,062**	**0,063**	-	-	0,040	**0,023**
deflate		**0,051**	**0,068**	-	-	**0,012**	**0,018**
myersdiff		0,135	0,128	-	-	0,022	0,056
levenshteindiff		0,088	0,118	-	-	0,089	0,073
idadiff	1	-	-	0,061	0,119	0,025	0,061
idadiffc		-	-	0,055	0,115	0,013	0,064
ghydradiff		-	-	0,063	0,113	0,025	0,059

ghydradiffc		-	-	0,305	0,216	0,220	0,119
ninjadiff		-	-	0,429	0,292	0,391	0,209
ninjadiffc		-	-	0,087	0,145	0,030	0,103

Веса в табл. 4 дополнительно подтверждают выводы исследования [9]: алгоритмы tlsh и sdhash показывают лучший результат (согласно табл. 4, дают больший вклад в принятое классификатором решение) при нахождении похожих файлов с исполнимым кодом, чем алгоритм ssdeep. Возможная причина этого заключается в том, что алгоритм, лежащий в основе ssdeep, зависит от длины входных данных: чем больше данных, тем длиннее хеш-значение. А так как применение разных компиляторов и разных опций компиляции к одному исходному коду приводит к созданию файлов разной длины, то и хеш-значения будут разной длины. И поэтому соответствующий алгоритм сравнения хеш-значений одинаковых по функциональности, но разных по размеру программ дает низкий уровень похожести.

Табл. 5. Влияние показателей на точность классификаторов
Table 5. Influence of features on the accuracy of classifiers

Классификатор	**Номер эксп-та**	**Исключенные показатели**				
		–	*ssdeep*	*ssdeep, bz2*	*ssdeep, bz2, lzma*	*ssdeep, bz2, lzma, deflate*
SVM	4	0,88	0,88	0,88	0,88	0,88
	5	0,93	0,94	0,94	0,94	0,94
DTC	4	0,91	0,91	0,91	0,91	0,9
	5	0,97	0,97	0,97	0,97	0,97
RFC	4	0,94	0,94	0,94	0,94	0,92
	5	0,98	0,98	0,98	0,98	0,98
LDA	4	0,87	0,87	0,86	0,86	0,86
	5	0,91	0,91	0,91	0,91	0,9
KNC	4	0,93	0,93	0,93	0,93	0,91
	5	0,98	0,98	0,98	0,98	0,98
LR	4	0,87	0,87	0,87	0,87	0,86
	5	0,91	0,91	0,91	0,91	0,9
GNB	4	0,65	0,65	0,65	0,67	0,67
	5	0,7	0,69	0,69	0,69	0,69

В табл. 5 приведены результаты четвертого и пятого экспериментов, которые отличаются от экспериментов 1 и 3 только тем, что исследовалось влияние показателей *ssdeep*, *bz2*, *lzma*, *deflate* на точность принимаемого классификатором решения, путем последовательного исключения этих показателей при обучении классификаторов. Перечисленные показатели выбраны в связи с тем, что в эксперименте 1 они обладают наименьшими весами, согласно табл. 4. В эксперименте 3 наименьшими весами обладают *ssdeep*, *lzma*, *deflate*, *tlsh*, однако показатель *tlsh* в четвертом и пятом экспериментах не исключался при обучении, так как, согласно первому эксперименту, он обладает высоким весом. Несмотря на то, что вес показателя *bz2* в третьем эксперименте больше веса *tlsh*, показатель *bz2* в четвертом и пятом экспериментах исключался. Это связано с тем, что другие два показателя похожести на основе коэффициента сжатия файлов (*lzma* и *deflate*) обладают малым весом, поэтому представляется обоснованным исследование влияния на решение классификатора всех показателей, построенных на основе коэффициентов сжатия файлов.

Серым цветом в табл. 5 выделены те ячейки, в которых наблюдается снижение уровня точности по сравнению со значением в ячейке из той же строки и предыдущего столбца: исключение из обучения показателя приводит к снижению уровня точности. Заметим, что для всех классификаторов, за исключением классификатора *GNB*, с наименьшим уровнем точности, исключение из обучения показателя *ssdeep* не уменьшает точности классификации, а для классификатора *SVM* даже увеличивает точность. Поэтому представляется, что значение хеш-функции ssdeep можно исключить из числа статических характеристик. К

нерекомендуемым к применению хеш-функциям, вероятно, стоит отнести и dcfldd [37], длина хеш-значения которой по аналогии с ssdeep, зависит от длины входного файла. Заметим, что дополнительное исключение всех показателей, основанных на степени сжатия файлов, приводит к уменьшению точности для всех классификаторов (хотя бы в одном из экспериментов 4 или 5), кроме *SVM* и *GNB*. Это может свидетельствовать в пользу применения таких показателей при обнаружении похожести файлов.

5. Заключение

Отличительной особенностью предлагаемого в работе способа сравнения файлов с исполнимым кодом является использование двух типов характеристик: универсальных и специализированных. Результаты показали, что наибольший вклад в принятие решения классификатором вносят специализированные характеристики, средний вес которых, как минимум в два раза, больше среднего веса универсальных характеристик. Тем не менее совместное применение характеристик разных типов повышает точность классификации, а наилучшей точностью классификации обладают классификаторы *KNeighborsClassifier* и *RandomForestClassifier*.

Начальное описание программы может быть расширено значениями других средств анализа, в том числе значениями других нечетких хеш-функций, не учитывающих формат данных, например, mvhash-b [38], а также, например, значениями хеш-функций, которые вычисляются по графу потока управления программы, таких, как machoce [39], machoke [40] (однако не для всех рассмотренных дизассемблеров имеются реализации этих алгоритмов). Разные алгоритмы сжатия также могут использоваться для увеличения размерности вектора показателей, однако, как показывают результаты для одного из лучших классификаторов – для *RFC* (см. табл. 4) – соответствующие показатели вносят наименьший вес в принятие решения (если не учитывать показатель ssdeep, вес которого минимален). Вектор показателей также может быть расширен за счет использования других средств сравнения файлов исполнимого кода, например, в дополнение к BinDiff может быть использовано средство BCC, предложенное в [41] и показавшее, согласно [20], наибольшую точность сравнения бинарного кода.

Дальнейшим направлением исследования является оптимизация набора характеристик, применение характеристик, получаемых средствами динамического анализа программ, например, средствами символьного анализа, исследование качества предлагаемого метода для оценки похожести программ из набора, отличного от coreutils, а также оценка качества обфусцирующих преобразований с помощью построенных классификаторов.

Авторы благодарят рецензентов за замечания, позволившие глубже взглянуть на проблему сравнения бинарных файлов, в частности, на проблему выбора обучающего множества пар похожих и непохожих программ.

Список литературы / References

[1] Нурмухаметов А.Р. Применение диверсифицирующих и обфусцирующих преобразований для изменения сигнатуры программного кода. Труды ИСП РАН, том 28, вып. 5, 2016 г., стр. 93-104 / Nurmukhametov A. R. The application of compiler-based obfuscation and diversification for program signature modification. Trudy ISP RAN/Proc. ISP RAS, 2016, vol. 28, issue 5, pp. 93-104 (in Russian). DOI: 10.15514/ISPRAS-2016-28(5)-5.

[2] Терешкин А.В. Разработка и исследование метода защиты от удаленных атак на основе диверсификации программного обеспечения. Диссертация на соискание степени кандидата технических наук. Южный федеральный университет, 2007 г., 157 стр. / Tereshkin A.V. Development and research of a method of protection against remote attacks based on software diversification. Dissertation for the degree of candidate of technical sciences. South Federal University, 2007, 157 p. (in Russian).

[3] Collberg C., Thomborson C. Watermarking, Tamper-Proofng, and Obfuscation – Tools for Software Protection, IEEE Transactions on Software Engineering, vol. 28, issue 8, 2002, pp. 735-746.
[4] Walenstein A., El-Ramly M. et al. Similarity in programs. In Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, vol. 6301, 2007, 8 p.
[5] Borisov P.D., Kosolapov Yu.V. On the automatic analysis of the practical resistance of obfuscating transformations. Automatic Control and Computer Sciences, vol. 54, issue 7, 2020, pp. 619-629.
[6] Holder W., McDonald J.T., Andel T.R. Evaluating optimal phase ordering in obfuscation executives. In Proc. of the 7th Software Security, Protection, and Reverse Engineering/Software Security and Protection Workshop, 2017, pp. 1-12.
[7] Borisov P.D., Kosolapov Yu.V. Similarity Features for The Evaluation Of Obfuscation Effectiveness. In Proc. of the International Conference on Decision Aid Sciences and Application (DASA), 2020, pp. 898-902.
[8] Борисов П.Д., Косолапов Ю.В. О характеристиках символьного исполнения в задаче оценки качества обфусцирующих преобразований. Моделирование и анализ информационных систем, том. 28, вып. 1, 2021 г., стр. 38-51. / Borisov P.D., Kosolapov Yu.V. On characteristics of symbolic execution in the problem of assessing the quality of obfuscating transformations. Modeling and Analysis of Information Systems, vol. 28, issue 1, 2021, pp. 38-51 (in Russian).
[9] Pagani F., Dell'Amico M., Balzarotti D. Beyond precision and recall: understanding uses (and misuses) of similarity hashes in binary analysis. In Proc. of the Eighth ACM Conference on Data and Application Security and Privacy, 2018, pp. 354-365.
[10] Ding S.H., Fung, B.C., Charland P. Asm2vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization. In Proc. of the IEEE Symposium on Security and Privacy (SP), 2019, pp. 472-489.
[11] Юмаганов А.С. Поиск похожих символьных последовательностей и графов на основе методов машинного обучения. Диссертация на соискание степени кандидата технических наук. Самарский национальный исследовательский университет имени академика С.П. Королева, 2020 г., 130 стр. / Yumaganov A.S. Search for similar character sequences and graphs based on machine learning methods. Dissertation for the degree of candidate of technical sciences. Samara National Research University named after Academician S.P. Korolev, 2020, 130 p. (in Russian).
[12] Саргсян С., Курмангалеев Ш. и др. Масштабируемый инструмент поиска клонов кода на основе семантического анализа программ. Труды ИСП РАН, том 27, вып. 1, 2015 г., стр. 39-50. / Sargsyan S., Kurmangaleev S. et al. Scalable code clone detection tool based on semantic analysis, Proceedings of ISP RAS, vol. 27, issue 1, 2015, pp. 39–50 (in Russian). DOI: 10.15514/ISPRAS-2015-27(1)-3.
[13] Осадчая А.О., Исаев И.В. Метод поиска клонов в программном коде. Научно-технический вестник информационных технологий, механики и оптики, том 20, вып. 5, 2020 г., стр. 714-721. / Osadchaya A.O., Isaev I.V. Search of clones in program code. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 20, issue 5, 2020, pp. 714-721 (in Russian).
[14] Bellon S., Koschke R. et al. Comparison and evaluation of clone detection tools. IEEE Transactions on Software Engineering, vol. 33, issue 9, 2007, pp. 577-591.
[15] Sarantinos N., Benzaid C. et al. Forensic malware analysis: The value of fuzzy hashing algorithms in identifying similarities. In Proc. of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2016, pp. 1782-1787.
[16] Oliver J., Cheng C., Chen Y. TLSH – a loclity sensitive hash. In Proc. of the Fourth Cybercrime and Trustworthy Computing Workshop, 2013, pp. 7-13.
[17] Mohsen R. Quantitative measures for code obfuscation security. PhD Thesis, Imperial College London, 2016, 216 p.
[18] BinExport. Available at: https://github.com/google/binexport, accessed 10.10.2022.
[19] BinDiff. Available at: https://www.zynamics.com/, accessed 10.10.2022.
[20] Arutunian M., Hovhannisyan H. et al. A Method to Evaluate Binary Code Comparison Tools. In Proc. of the 2021 Ivannikov Memorial Workshop (IVMEM), 2021, pp. 3-5.
[21] A powerful disassembler and a versatile debugger. Available at: https://hex-rays.com/ida-pro/, accessed 10.10.2022.
[22] Ghidra Software Reverse Engineering Framework. Available at: https://github.com/NationalSecurityAgency/ghidra, accessed 10.10.2022.
[23] Binary Ninja. Available at: https://binary.ninja/, accessed 10.10.2022.
[24] Kornblum J. Identifying almost identical files using context triggered piecewise hashing. Digital investigation, vol. 3, 2006, pp. 91-97.

[25] Roussev V. An evaluation of forensic similarity hashes. Digital investigation, vol. 8, 2011, pp. S34-S41.
[26] Pavlov I. Lzma SDK (software development kit). Available at: https://7-zip.org/sdk.html, accessed 10.10.2022.
[27] Seward, Julian. bzip2 and libbzip2. Avaliable at http://www. bzip. org, accessed 10.10.2022.
[28] Oswal S., Singh A., Kumari K. Deflate compression algorithm. International Journal of Engineering Research and General Science, vol. 4, issue 1, 2016, pp. 430-436.
[29] Radare2. Available at: https://rada.re/, accessed 10.10.2022.
[30] Myers E. W. An O(ND) difference algorithm and its variations. Algorithmica, vol. 1, issue 1, 1986, pp. 251-266.
[31] В.И. Левенштейн. Двоичные коды, способные исправлять удаления, вставки и обращения. Доклады Академии Наук СССР, том 163, no. 4, 1966 г., стр. 845-848. / V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Soviet physics doklady, vol. 10, no. 8, 1966, pp. 707-710.
[32] Coreutils – GNU core utilities. Available at: https://www.gnu.org/software/coreutils/, accessed 10.10.2022.
[33] GCC, the GNU Compiler Collection. Available at: https://gcc.gnu.org/, accessed 10.10.2022.
[34] Clang: a C language family frontend for LLVM. Available at: https://clang.llvm.org/, accessed 10.10.2022.
[35] AMD Optimizing C/C++ and Fortran Compilers (AOCC). Available at: https://developer.amd.com/amd-aocc/, accessed 10.10.2022.
[36] Scikit-learn. Machine learning in Python. Available at: https://scikit-learn.org/, accessed 10.10.2022.
[37] Dcfldd. Available at: https://dcfldd.sourceforge.net/, accessed 10.10.2022.
[38] Breitinger F., Astebol K.P. et al. mvhash-b – A new approach for similarity preserving hashing. In Proc. of the Seventh International Conference on IT Security Incident Management and IT Forensics, 2013, pp. 33-44.
[39] Machoc hash. Available at: https://github.com/ANSSI-FR/polichombr/blob/dev/docs/MACHOC_HASH.md, accessed 10.10.2022.
[40] Machoke. Available at: https://github.com/conix-security/machoke, accessed 10.10.2022.
[41] Aslanyan H., Avetisyan A. et al. Scalable Framework for Accurate Binary Code Comparison. In Proc. of the 2017 Ivannikov ISPRAS Open Conference (ISPRAS), 2017, pp. 34-38.

Информация об авторах / Information about authors

Петр Дмитриевич БОРИСОВ является заведующим лабораторией в ФГАНУ НИИ «Спецвузавтоматика». Сфера научных интересов: исследование программного кода, теоретическая и практическая обфускация кода.

Petr Dmitrievich BORISOV is the head of the laboratory of the FGANU Research Institute "Spetsvuzavtomatika". Research interests: code analisys, theoretical and practical code obfuscation.

Юрий Владимирович КОСОЛАПОВ – кандидат технических наук, доцент кафедры алгебры и дискретной математики института математики, механики и компьютерных наук им. И.И. Воровича. Сфера научных интересов: помехоустойчивые коды в криптографии, теоретическая и практическая обфускация кода.

Yuri Vladimirovich KOSOLAPOV – Candidate of Technical Sciences, Associate Professor, Department of Algebra and Discrete Mathematics, Institute of Mathematics, Mechanics and Computer Science named after I.I. Vorovich. Research interests: error-correcting codes in cryptography, theoretical and practical code obfuscation.

DOI: 10.15514/ISPRAS-2022-34(5)-5

Библиотека для разработки компиляторов

С.В. Миронов, ORCID: 0000-0003-3699-5006 <mironovsv@sgu.ru>
И.А. Батраева, ORCID: 0000-0002-6539-8473 <batraevaia@info.sgu.ru>
П.Д. Дунаев, ORCID: 0000-0002-9142-0945 <herrpaulvondonau@outlook.com>

Саратовский государственный университет,
410012, Россия, г. Саратов, ул. Астраханская, 83

Аннотация. Работа посвящена разработке библиотеки, предназначенной для реализации компиляторов. Статья содержит описание возможностей библиотеки и основных принципов её функционирования. В ходе работы была изучена и реализована генерация синтаксических анализаторов с помощью LR(1)-автоматов, были спроектированы и реализованы два вспомогательных языка: язык запросов к семантической сети и язык, предназначенный для генерации исполняемого кода. Результатом работы является библиотека для платформы .NET (библиотека тестировалась, в частности, для языка C#), которая содержит классы, существенно облегчающие реализацию синтаксического анализа исходного кода, семантического анализа и генерацию исполняемого файла. Данная библиотека не имеет внешних зависимостей, кроме стандартной библиотеки .NET.

Ключевые слова: библиотека; компилятор; генерация синтаксических анализаторов; кодогенерация; **.**NET

Для цитирования: Миронов С.В., Батраева И.А., Дунаев П.Д. Библиотека для разработки компиляторов. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 77-88. DOI: 10.15514/ISPRAS-2022-34(5)-5

Library for Development of Compilers

S.V. Mironov, ORCID: 0000-0003-3699-5006 <mironovsv@sgu.ru>
I.A. Batraeva, ORCID: 0000-0002-6539-8473 <batraevaia@info.sgu.ru>
P.D. Dunaev, ORCID: 0000-0002-9142-0945 <herrpaulvondonau@outlook.com>

Saratov State University,
83 Astrakhanskaya Street, Saratov, Russia, 410012

Abstract. This work is devoted to the development of a library designed to implement compilers. The article contains a description of the library's features and the main points of its functioning. In the course of the work, the generation of parsers using LR(1)-automata was studied and implemented, two auxiliary languages were designed and implemented: a semantic network query language and a language designed to generate executable code. The result of the work is a library for the platform .NET (the library was tested for the C# language), which contains classes that make easier the implementation of source code parsing, semantic analysis, and executable file generation. This library does not have external dependencies, except for the standard .NET library.

Keywords: library; compiler; parsers generation; code generation; .NET

For citation: Mironov S.V., Batraeva I.A., Dunaev P.D. Library for Development of Compilers. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 77-88 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-5

1. Введение

Потребность в новых языках программирования постоянно растёт. Разработчики могут столкнуться с необходимостью создать предметно-ориентированный язык. Некоторым командам может потребоваться разработать свой компилятор языка общего назначения или собственного его диалекта. Энтузиасты продолжают воплощать свои идеи, создавая новые языки программирования. Разработка компилятора является сложной задачей, часть решения которой может быть вынесена в качестве библиотечных функций для использования в произвольном количестве проектов. Цель данной работы – предложить новый отечественный открытый продукт, позволяющий упростить и ускорить их разработку. Библиотека призвана стать наиболее удобным инструментом для быстрой и лаконичной реализации предметно-ориентированных языков в рамках промышленной разработки и для начинающих разработчиков компиляторов.

Библиотека[1] реализована на языке C#, она не имеет иных зависимостей, кроме стандартной библиотеки .NET. Использование библиотеки возможно на любых языках для платформы .NET 6.0 и выше. Таким образом, компиляторы, использующие библиотеку, являются кроссплатформенными, однако генерировать исполняемые файлы через библиотечные функции в момент написания статьи можно пока только для 64-битных операционных систем семейства Windows.

2. Анализ существующих решений

Функциональность библиотеки можно разделить на три следующих блока.

1) Синтаксический анализ. Библиотека содержит классы для конструирования и использования парсеров. Грамматика языка описывается в пользовательских классах с помощью атрибутов – особых конструкций .NET-языков, снабжающих различные элементы метаданных дополнительной информацией, которыми помечаются, в данном случае, методы-обработчики продукций. Обработчик должен возвращать задаваемое пользователем внутреннее представление описываемой продукцией конструкции;
2) Семантическая сеть. В библиотеке предусмотрены классы для описания отношений между сущностями кода (такими как классы, методы, переменные и т. д.) и решения задачи поиска объекта, связанного более сложными отношениями с данным (например, отношение доступности между областью видимости и переменной, полем или классом). Такой поиск кодируется с помощью встроенного в библиотеку языка запросов.
3) Генерация исполняемого кода. В библиотеку встроен специальный язык кодогенерации. Это процедурный язык, который реализован “поверх” .NET-языка, на котором реализуется компилятор, т.е. его конструкции представляют собой вызовы методов определённых в библиотеке классов, включая перегруженные операторы.

Первый блок может быть реализован с помощью генерации различных типов распознавателей. В частности, существуют такие распознаватели, как LL [1], LR [2] и LALR [3], в том числе их обобщённые (generalized) модификации (GLL, GLR) [4], способные обрабатывать любые контекстно-свободные грамматики, толерантные модификации [5], предназначенные для распознавания определённых, интересных в рамках конкретных задач участков кода, а также “ленивые” модификации [6], которые позволяют исключить временные затраты на разбор неиспользуемых определений языка. Библиотека генерирует классические LR(1)-автоматы, поскольку они имеют большую распознавательную способность.

В качестве аналогов первого блока можно выделить GNU Bison [7], ANTLR [8] и Coco/R [9]. С их помощью анализатор строится следующим образом: разработчик описывает грамматику с помощью специального языка, этот код преобразуется генератором в парсер для целевого

[1] https://github.com/herrpaulvd/CompileLib

языка, который, в конечном счёте, используется в проекте компилятора. Наше решение требует определённым образом описанные классы, которые анализируются во время выполнения с помощью рефлексии. Преимуществом нашего подхода является быстрая скорость внесения изменений в описание парсера, в то время как при использовании аналогов после корректировки описания грамматики требуется повторно генерировать парсер, заменяя им результат старой генерации. В качестве недостатка выступают затраты времени исполнения на построение синтаксического анализатора при каждом запуске компилятора, что частично решается кэшированием построенного распознавателя. Описанные преимущества и недостатки делают наш генератор парсеров более подходящим для языков, от компилятора которых не требуется высокой производительности, например, некоторых предметно-ориентированных языков, и менее подходящим для компиляторов языков с богатым синтаксисом, таких как, например, C++.

Аналогом второго блока выступает продвинутый инструмент, описанный в статье [10] и позволяющий устанавливать связи между сущностями программ на языках C и C++. Наша библиотека предлагает инструментарий для решения этой задачи для любого языка и для произвольных связей, которые достаточно описать с помощью декларативного предметно-ориентированного языка, близкого к языку теории множеств.

Третий блок может быть реализован двумя способами: с помощью высокоуровневых виртуальных машин, таких как LLVM [11] или System.Reflection.Emit [12], или генерации кода на другом языке, например, на C или языке ассемблера, отправляя результат в соответствующий компилятор или ассемблер. В библиотеке для генерации кода предложен свой язык, конструкции которого представляют собой вызовы библиотечных методов. Синтаксис языка является более сложным, чем просто последовательность инструкций – язык допускает составные выражения, компонентами которых могут быть арифметические операции, вызовы функций и индексация указателей как в C. Такой подход перенимает преимущества генерации кода на другом высокоуровневым языке, исключая временные затраты на синтаксический анализ сгенерированного кода (он фактически происходит во время компиляции кода самого компилятора).

В отличие от всех вышеперечисленных аналогов, библиотека является комплексным решением, включающим в себя инструменты для всех компонентов компилятора, хотя возможно использование блоков библиотеки по-отдельности. Целостные решения являются обычно более предпочтительными при отсутствии специфических требований, поскольку позволяют сэкономить время на подбор подходящей комбинации фреймворков. Все блоки объединены в одну DLL, что значительно упрощает использование библиотеки: для её использования достаточно добавить в проект всего одну DLL, не производя дополнительных настроек. Таким образом, библиотека подходит, в первую очередь, для быстрой и простой разработки компиляторов, что делает её использование удобным в основном в рамках двух сценариев:

1) реализация предметно-ориентированных языков при отсутствии необходимости обеспечить высокую производительность компилятора или исполняемого кода;
2) первые шаги в создании собственных компиляторов, когда важно, в первую очередь, получить представление о самом процессе разработки компилятора, о его компонентах, не углубляясь при этом в детали, и закрепить изученное, разработав свой “игрушечный” компилятор.

3. Возможности библиотеки

В этом разделе описаны основные возможности библиотеки и некоторые идеи, использованные для их реализации.

3.1 Лексический и синтаксический анализ

Для конструирования парсера предназначен класс `ParsingEngineBuilder`. Он содержит метод, принимающий в качестве параметров идентификатор лексемы, регулярное выражение стандарта POSIX [13] и произвольное количество определяемых пользователем классов символов, передаваемые как замыкания вида `char -> bool`.

В качестве распознавателя лексем применяются недетерминированные конечные автоматы (НКА), модифицированные в целях экономии памяти. Использование своего варианта НКА мотивировано тем, что реализации обычных НКА, построенных на основе регулярных выражений, должны хранить множество однотипных переходов. Например, регулярному выражению `[[:print:]]*` соответствует НКА с одним состоянием, из которого существует n переходов в себя же, где n – количество печатных символов, которых, например, в Unicode более 60000. Необходим способ, который бы уменьшал количество хранимых переходов.

Основная идея модификации НКА – объединить некоторые группы символов в один, так чтобы автомат работал так, как если бы ему на вход вместо обычных символов поступали "объединённые". Если обычные НКА представлены пятёркой $(Q, \Sigma, \delta, q_0, F)$, где Q – множество состояний автомата, Σ – входной алфавит, $\delta: Q \times \Sigma \longrightarrow 2^Q$ – функция переходов, q_0 – начальное состояние, $F \subseteq Q$ – множество конечных состояний, то в нашем представлении автомат описывается семёркой, в которую дополнительно входят U – промежуточный алфавит, $\gamma: \Sigma \longrightarrow 2^U$ – функция неоднозначного преобразования символов входного алфавита в символы промежуточного, причём изменён тип функции переходов $\delta: Q \times \mathrm{U} \longrightarrow Q$.

На каждой итерации автомат сначала считывает очередной входной символ, затем преобразует его в символ промежуточного алфавита (этот шаг недетерминирован), после этого в зависимости от полученного символа и текущего состояния автомат переводится в новое состояние (этот шаг детерминирован, в отличие от аналогичного шага в обычных НКА). Нетрудно показать, что каждый обычный НКА можно преобразовать в модифицированный, и наоборот, т. е. мощности этих типов распознавателей совпадают.

Использовать модифицированный НКА предлагается следующим образом. В качестве промежуточного алфавита используется множество всех вхождений одиночных символов и выражений в квадратных скобках для преобразуемого регулярного выражения. Например, для выражения `_[[:alnum:]]+_[01]?` промежуточный алфавит будет состоять из 4 символов: первое вхождение символа `'_'`, скобочное выражение `[[:alnum:]]`, представляющее множество всех букв и цифр, второе вхождение символа `'_'`, выражение `[01]`. Функция γ не строится – она будет определена неявно: каждый символ входного алфавита будет переводиться во все те символы промежуточного алфавита, которые представляют регулярные подвыражения, которым удовлетворяет этот входной символ.

Прочие элементы строятся согласно алгоритму построения обычного НКА по регулярному выражению [14], как если бы на вход было передано исходное регулярное выражение, в котором одиночные символы и выражения в квадратных скобках заменены на соответствующие символы промежуточного алфавита. При этом в результате применения алгоритма [14] может получиться так, что для некоторых $q \in Q$ и $u \in U$ функция переходов возвращает несколько значений, в то время как, согласно определению модифицированного НКА, значение функции должно быть определено однозначно. В качестве выхода из этой ситуации можно заменить символ u на n символов $u_1, u_2, \cdots, u_n$, где n – количество переходов, каждому переходу, таким образом, будет соответствовать свой символ промежуточного алфавита, каждый символ представляет то же регулярное подвыражение, что и u.

Каждый промежуточный символ будет представлен предикатом вида `char -> bool`, который определяет, удовлетворяет ли аргумент данному регулярному подвыражению, а

список переходов будет храниться в виде массива, в котором каждому индексу-состоянию i будет соответствовать список пар вида (p, j), где p – предикат, j – новое состояние. НКА будет работать по следующему алгоритму:

1) автомат начинает работу в множестве состояний $Q_{curr} = \{q_0\}$; функция переходов описана массивом T, в котором каждому состоянию q_i сопоставлено множество $T[q_i] = \{(p_{i,j}, q'_{i,j}) \mid j = \overline{1, k_i}\}$;
2) считывается входной символ c;
3) множество состояний заменяется на новое: $Q_{new} \coloneqq \{q' \mid (p, q') \in T[q], q \in Q_{curr}, p(q)$ истинно$\}$; $Q_{curr} \coloneqq Q_{new}$.

Для регулярного выражения `[[:print:]]*`, рассмотренного выше, будет храниться массив из одного элемента (поскольку состояние только одно), этим элементом будет список из одной пары: первым элементом пары будет предикат $p(c) = (20_{16} \leq c \leq 7E_{16}) \vee (c \geq 100_{16})$, который проверяет, является ли символ печатным в Unicode, вторым – единственное состояние автомата.

Синтаксис языка задаётся с помощью методов-обработчиков продукций, сами продукции задаются с помощью атрибутов, которыми помечается метод и его параметры. Отсутствует возможность задать не контекстно-свободную грамматику. Метод помечается атрибутом `SetTag`, в котором указывается идентификатор нетерминала, находящегося в левой части продукции. Каждому параметру, кроме, возможно, последнего, соответствует как минимум один символ продукции. Порядок параметров совпадает с порядком символов в продукции. Последний параметр может быть зарезервирован для обработчика ошибок.

Параметр может быть помечен одним из атрибутом повторения. С их помощью можно упростить описание грамматики. Например, если предполагается, что один элемент в некоторой конструкции может отсутствовать, можно использовать атрибут `Optional`, определив одну продукцию вместо нескольких.

Каждый параметр, кроме, возможно, последнего, должен быть помечен

ровно одним атрибутом, задающим символ грамматики, и не более чем одним

атрибутом, задающим повторение символов.

Символ грамматики могут задавать следующие атрибуты.

- `RequireTags(params string[] tags)` задаёт множество допустимых символов (лексем или нетерминалов). Указание нескольких символов позволяет избежать определения нескольких продукций для языковой конструкции, которая предполагает использование в одинаковом качестве разнотипных элементов. Например, в качестве элементов выражений могут выступать как числовые константы, так и идентификаторы.
- `Keywords(params string[] keywords)` задаёт множество допустимых ключевых слов (здесь под ними понимаются лексемы, которые задают единственную строку и не имеют собственного идентификатора. Ключевые слова передаются в качестве параметром атрибута “как есть”, т. е. они интерпретируются как обычные строки, но не регулярные выражения).

Повторение символов задаётся следующими атрибутами.

- `Single` – символ повторяется ровно один раз. Если атрибут повторения не указан, применяется по умолчанию;
- `Optional(bool greedy)` – символ является необязательным, т. е. может отсутствовать. Если параметр `greedy` установлен в `true`, то при наличии символа он обязательно будет прочитан. Если символ не прочитан, значение параметра устанавливается в `null`;
- `Many(bool canBeEmpty)` – символ может повторяться более одного раза. Если `canBeEmpty = true`, символ может ни разу не повториться. Помеченный этим

атрибутом параметр, а также следующие за ним, помеченные `TogetherWith`, должны иметь тип массива. Длина массива есть количество прочитанных повторений;

- `TogetherWith` —удлиняет цепочку символов, которая попадает под модификатор повторения. Первый параметр не может быть помечен данным атрибутом. Например, если i-й параметр и идущий перед ним помечены данным атрибутом, а $(i-2)$-й параметр помечен атрибутом `Optional`, то все три символа будут одновременно прочитаны или не прочитаны.

В качестве примера рассмотрим сигнатуру обработчика оператора `if` в С-подобных языках, код представлен на C#:

```
[SetTag("statement")]
public static Statement? ReadIfStatement(
    [Keywords("if")] Parsed<string> kw,
    [Keywords("(")] string brOpen,
    [RequireTags("expr")] Expression condition,
    [Keywords(")")] string brClose,
    [RequireTags("statement")] Statement ifBranch,
    [Optional(true)][Keywords("else")] string kwElse,
    [TogetherWith][RequireTags("statement")] Statement elseBranch)
```

Здесь `Expression` и `Statement` – пользовательские классы, определяющие конструкции языка. Ключевое слово `else` помечено атрибутом `Optional(true)`, за ним идёт ветка, помеченная `TogetherWith`. Поскольку параметр `greedy = true`, ветка `else`, следующая за остальной частью оператора, будет считываться и присоединяться к оператору. Таким образом, атрибут `Optional(true)` решает, в частности, проблему “висящего else”, описанную в [15], избавляя разработчика от необходимости разбивать описание оператора на несколько продукций.

Последний параметр обработчика может быть зарезервирован для обработки ошибок. Он помечается атрибутом `ErrorHandler` и имеет тип `ErrorHandlingDecider`. При отсутствии ошибок параметр имеет значение `null`, иначе представляет объект, с помощью которого принимается решение об обработке ошибки. При обнаружении недопустимой лексемы обработчик находит подходящую конструкцию, структура которой, возможно, была нарушена, и обработчик которой имеет соответствующий последний параметр. Обработчик, получив информацию о данной лексеме и о уже прочитанной части конструкции, принимает решение о том, как лексема должна быть обработана: например, проигнорирована, заменена, передана другому обработчику, или нужно остановить весь анализ. Принятие решения осуществляется через вызов соответствующих методов объекта `ErrorHandlingDecider`.

Класс, содержащий обработчики, передаётся объекту `ParsingEngineBuilder`. После того, как полностью были заданы лексика и синтаксис языка, вызывается метод, конструирующий объект класса `ParsingEngine`, содержащий набор автоматов для проведения лексического анализа и LR(1)-автомат, осуществляющий синтаксический анализ. В случае, если грамматика не является LR(1)-грамматикой, метод выбрасывает исключение, содержащее подробную информацию о возможных неоднозначностях.

3.2 Семантическая сеть и язык запросов

Библиотека предоставляет возможность организовывать такие элементы кода, как классы, методы, переменные и т. д. в семантическую сеть и писать к этой сети запросы. Объект семантической сети должен быть экземпляром класса-наследника `CodeObject`. Объекту должно быть задано имя (не обязательно уникальное) и тип, кроме того, он имеет две встроенные коллекции. Первая коллекция содержит атрибуты объекта, заданные в виде пар имя атрибута – значение атрибута, вторая – отношения с другими объектами, заданные в виде пар имя отношения – объект. Отношения, хранящиеся в данной коллекции, являются

простыми, как, например, отношения между классом и его членом. При решении задач, связанных с семантическим анализом, требуется устанавливать более сложные взаимосвязи, такие как, например, поиск объекта с заданным именем, видимого в заданной области. Для описания подобных сложных отношений был разработан специальный язык запросов.

Скрипт на этом языке состоит из правил – конструкций, которые описывают сложные отношения. Каждое правило имеет имя, набор явных параметров, ровно один неявный параметр и тело. Неявным параметром всегда является объект, относительно которого начинается поиск.

Правило имеет вид `@name(params) = expression;` где `name` – имя правила, `params` – имена явных параметров, указанных через запятую, `expression` – тело правила. Последнее является выражением, конструируемым по следующим правилам:

1) `name(obj_name)` – представляет собой множество объектов, связанных с данным каким-либо отношением и имеющих имя `obj_name`;
2) `type(obj_type)` – множество объектов, связанных с данным каким-либо отношением и имеющих тип `obj_type`;
3) `relation(relation_name)` – множество объектов, связанных с данным отношением `relation_name`;
4) `attribute(attr_name)` – множество объектов, связанных с данным каким-либо отношением и имеющих атрибут `attr_name` с любым значением;
5) `attribute(attr_name, attr_value)` – множество объектов, связанных с данным каким-либо отношением и имеющих атрибут `attr_name` со значением `attr_value`;
6) `@rule(params)` – множество объектов, получаемое запуском правила `rule` с заданными явными параметрами и данным объектом в качестве неявного;
7) Если `X` и `Y` – выражения языка запросов, то
   a) `(X) & (Y)` – пересечение множеств, представленных выражениями;
   b) `(X) | (Y)` – объединение множеств;
   c) `(X) ?| (Y)` возвращает `X`, если `X` не пусто, иначе `Y`;
   d) `(X) . (Y)` – вычисляется множество `X`, к каждому элементу которого как к неявному параметру применяется выражение `Y` как самостоятельное правило, результаты всех применений выражения `Y` объединяются в одно множество, возвращаемое данным выражением;
   e) `(X) +. (Y)` аналогично `(X) | ((X) . (Y))`;
   f) Круглые скобки могут быть опущены в соответствии со следующим приоритетом операций: наивысший приоритет – `&`, средний – `.` и `.+`, низший – `|` и `?|`;
8) Другие выражения недопустимы.

Правила могут быть рекурсивными. Скрипт передаётся в виде строки объекту класса `SemanticNetwork`, в котором содержатся методы для вызова правил по имени правила и объекту – неявному параметру. Ниже приведён пример правила для поиска объектов в локальной области видимости по заданному имени:

```
@local-search(@name) =
   name(@name) & type(local-var) #1
   ?| relation(parent) & type(scope) . @local-search(@name) #2
   ?| relation(parent) & type(method)
      . name(@name) & relation(parameter) #3
   ?| relation(parent) & type(method) & attribute(static)
      . relation(parent)
      . @class-search(@name, static, private) #4
   ?| relation(parent) & type(method) & attribute(instance)
      . relation(parent) . @class-search(@name, anymember,
```

```
    private) #5
  ?| relation(parent) & type(method) . @global-search(@name); #6
```

В качестве неявного параметра выступает локальная область видимости. Поиск происходит в несколько этапов (номера отмечены однострочными комментариями #), переход к следующему этапу происходит только если на предыдущем ничего не было найдено:

1) правило пытается найти переменные, находящиеся непосредственно в данной области видимости;
2) если таких нет, и эта область вложена ещё в одну такую же, поиск рекурсивно переходит в эту область;
3) если непосредственным предком области является метод, проверяются параметры метода;
4) если метод статический, передаётся управление другому правилу – `@class-search`, которое запускает поиск статических членов класса с заданным именем;
5) если метод экземплярный, передаётся управление правилу `@class-search`, которое запускает поиск любых членов класса с заданным именем;
6) происходит поиск в глобальной области видимости с помощью правила `@global-search`.

Как было показано выше, это правило использует два других пользовательских правила: `@class-search` и `@global-search`. Последнее принимает на вход только имя искомого объекта, который должен находиться в глобальной области видимости. Правило `@class-search` имеет 3 параметра: имя искомого объекта, значение атрибута, определяющего, является ли член класса статическим (`static`), экземплярным (`instance`), или должны рассматриваться оба варианта (`anymember`). Третий параметр определяет минимальную допустимую доступность члена класса (`private < protected < public`).

Предполагается, что семантическая сеть построена так, что каждый член класса обязательно имеет атрибут `anymember`, ровно один из атрибутов `static` или `instance`, а также набор атрибутов доступности, максимальным из которых является атрибут, соответствующий заданному в исходном коде модификатору, минимальным – `private`, а также представлены все атрибуты, находящиеся между ними в иерархии доступности. Так, публичные члены класса будут иметь все три атрибута доступности, protected-члены – `protected` и `private`, приватные члены – только `private`. Таким образом, избегается необходимость дублировать вызов `@class-search` для всех модификаторов доступа, достаточно лишь указать минимальный требуемый. В примере требуется поиск членов класса с любым модификатором доступа, поэтому в качестве параметра указан наименьший из них – `private`.

3.3 Язык генерации исполняемого кода

Это процедурный язык программирования, кодирование на котором осуществляется с помощью вызова соответствующих методов библиотеки. Он поддерживает различные типы, в числе которых есть:

1) целочисленные, знаковые и беззнаковые, размером в 1, 2, 4 и 8 байт;
2) вещественные, по 4 и 8 байт;
3) аналог типа `void`, используемый только для объявления функций, не предусматривающих возврата значения;
4) указатели;
5) структуры

При задании структур существует возможность указать выравнивание полей, так чтобы смещение каждого нового поля было кратно этому значению (аналогично `#pragma pack(n)` в C).

В языке можно определять собственные функции и импортировать функции из сторонних DLL, например, из системных kernel32.dll или user32.dll. Помимо функций, в языке можно объявлять глобальные и локальные переменные, константы, а также блоки инициализированных данных. Предусмотрены различные типы бинарных и унарных операций, допустима арифметика указателей. Указатели также могут быть индексированы.

Помимо функций, в качестве конструкций, управляющих потоком выполнения программы, в языке предусмотрены метки и операторы безусловного и условного переходов.

Ниже приведён участок кода, который определяет аналог функции `malloc` через системную функцию `HeapAlloc` (основной язык: C#) [16]:

```
compiler.OpenEntryPoint();
var hHeap = compiler.AddGlobalVariable(ELType.PVoid);
var GetProcessHeap = compiler.ImportFunction("kernel32.dll",
    "GetProcessHeap", ELType.PVoid);
var HeapAlloc = compiler.ImportFunction("kernel32.dll",
    "HeapAlloc", ELType.PVoid, ELType.PVoid,
    ELType.UInt32,ELType.UInt64);
hHeap.Value = GetProcessHeap.Call();
var malloc = compiler.CreateFunction(ELType.PVoid, ELType.UInt64);
var pSize = malloc.GetParameter(0);
malloc.Open();
compiler.Return(HeapAlloc.Call(hHeap,compiler.MakeConst(0U),
    pSize));
```

Объект `compiler` класса `ELCompiler` используется для определения различных объектов, таких как функции или глобальные переменные, а также для использования различных операторов или конструкций. В первой строке происходит обращение к точке входа – предопределённой функции. Во второй строке добавляется глобальная переменная `hHeap`, в которой будет сохранён дескриптор кучи, он имеет тип `void*`. Далее импортируются функции `GetProcessHeap` и `HeapAlloc` из kernel32.dll, предназначенные для получения дескриптора кучи и выделения области памяти в куче соответственно. Затем вызывается функция `GetProcessHeap`, возвращённое ей значение присваивается переменной `hHeap`. Далее определяется пользовательская функция `void* malloc(UINT64)`, и она же открывается на запись. В функцию добавляется единственное выражение – возврат результата вызова `HeapAlloc`, которой в качестве первого аргумента передаётся дескриптор кучи, в качестве второго – пустое множество флагов в виде 32-битного беззнакового нуля, в качестве последнего – размер выделяемой памяти.

Предполагается, что большая часть кода на языке кодогенерации будет расположена небольшими фрагментами между другими частями кода компилятора, также отвечающими за кодогенерацию. В качестве примера подобного случая приведём компиляцию оператора `if` (основной язык: C#):

```
var elselabel = compiler.DefineLabel(); // (1)
var endlabel = compiler.DefineLabel(); // (1)
var expr = Condition.CompileRight(compilation); // (2)
var texpr = Condition.Type;
if (!texpr.IsIntegerType())
    throw new CompilationError($"Invalid condition type
{texpr.Show(name2class)}", Line, Column);
compiler.GotoIf(!expr, elselabel); // (1)
CodeObject ifscope = new("", "scope", -1, -1);
ifscope.AddRelation("parent", compilation.Scope);
```

```
IfBranch.Compile(compilation.WithScope(ifscope)); // (2)
compiler.Goto(endlabel); // (1)
compiler.MarkLabel(elselabel); // (1)
if(ElseBranch is not null)
{
    CodeObject elsescope = new("", "scope", -1, -1);
    elsescope.AddRelation("parent", compilation.Scope);
    ElseBranch.Compile(compilation.WithScope(elsescope)); // (2)
}
compiler.MarkLabel(endlabel); // (1)
```

Здесь комментариями (1) отмечены строки, в которых расположены выражения языка кодогенерации, комментариями (2) отмечены вызовы методов, которые содержат другие выражения языка.

Сгенерированный код преобразуется в машинный код процессоров архитектуры AMD64, для написания библиотеки использовалась официальная документация [17], в качестве формата исполняемого файла был выбран Portable Executable, структура которого подробно описана в [18]. Генератор исполняемого кода не использует сторонних библиотек (кроме стандартной библиотеки .NET), ассемблеров или компоновщиков.

4. Заключение

В статье была рассмотрена функциональность библиотеки для создания компиляторов и ряд идей, использовавшихся при её написании, в частности, идея модификации определения недетерминированного конечного автомата для более компактного хранения распознавателя в оперативной памяти.

Были рассмотрены три возможности библиотеки: средства описания синтаксиса языка и синтаксического анализа, классы для построения семантической сети и язык запросов к ней, язык генерации исполняемого кода.

В качестве примера для тестирования на языке C# был разработан компилятор C-подобного языка, частично поддерживающего объектно-ориентированное программирование. Компилятор использовал все три функциональных блока библиотеки. Другие библиотеки, кроме описанной и стандартной, не были включены в проект.

Список литературы / References

[1] Lewis P., Stearns R. Syntax-Directed Transduction. Journal of the ACM, vol. 15, issue 3, 1968, pp. 465-488.

[2] Knuth D. On the Translation Languages from Left to Right. Information and Control, vol. 8, issue 6, 1965, pp. 607-639.

[3] DeRemer F. Practical Translators for LR(k) Languages. PhD Thesis, Massachusetts Institute of Technology, 1969, 215 p.

[4] Lang B. Deterministic Techniques for Efficient Non-Deterministic Parsers. Lecture Notes in Computer Science, vol. 14, 1974, pp. 255-269.

[5] Goloveshkin A.V. Tolerant parsing using modified LR(1) and LL(1) algorithms with embedded “Any” symbol. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 3, 2019, pp. 7-28. DOI: 10.15514/ISPRAS-2019-31(3)-1.

[6] Савицкий В.О., Сидоров Д.В. «Ленивый» анализ исходного кода на языках C и C++. Труды ИСП РАН, том 23, 2012 г., стр. 133-142 / Savitsky V.O., Sidorov D.V. Lazy source code analysis for C/C++ languages. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 133-142 (in Russian). DOI: 10.15514/ISPRAS-2012-23-8.

[7] GNU Bison. Available at: https://www.gnu.org/software/bison/.

[8] ANTLR. Available at: https://www.antlr.org/.

[9] Coco/R. Available at: https://ssw.jku.at/Research/Projects/Coco/.

[10] Белеванцев А.А., Велесевич Е.А. Анализ сущностей программ на языках Си/Си++ и связей между ними для понимания программ. Труды ИСП РАН, том 27, вып. 2, 2015 г., стр. 53-64. / A. Belevantsev,

E. Velesevich. Analyzing C/C++ Code Entities and Relations for Program Understanding Trudy ISP RAN /Proc. ISP RAS, vol. 27, issue 2, 2015, pp. 53-64 (in Russian). DOI: 10.15514/ISPRAS-2015-27(2)-4.

[11] The LLVM Compiler Infrastructure. Available at: https://llvm.org/.

[12] Microsoft Docs: System.Reflection.Emit Namespace. Available at: https://learn.microsoft.com/en-us/dotnet/api/system.reflection.emit?view=netstandard-2.0.

[13] The Open Group Base Specifications Issue 7, 2018 edition, IEEE Std 1003.1™-2017 (Revision of IEEE Std 1003.1-2008). Chapter 9. Regular Expressions. Available at: https://pubs.opengroup.org/onlinepubs/9699919799.

[14] Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Том 1: Синтаксический анализ. М., Издательство МИР, 1978, 616 стр. / Aho A., Ullman J. The Theory of Parsing, Translation and Compiling. Volume 1: Parsing. Prentice Hall, 1972, 542 p.

[15] Ахо А., Лам М. и др. Компиляторы. Принципы, технологии и инструментарий. М., Издательский дом Вильямс, 2008, 1178 стр. / Aho A., Lam M. et al. Principles, Techniques, & Tools. M., Addison Wesley, 2006, 1040 p.

[16] Microsoft Docs: HeapAlloc function (heapapi.h). Available at: https://learn.microsoft.com/en-us/windows/win32/api/heapapi/nf-heapapi-heapalloc.

[17] AMD64 Architecture Programmer's Manual: Volumes 1-5. Available at: https://www.amd.com/system/files/TechDocs/40332_4.05.pdf.

[18] Microsoft Portable Executable and Common Object File Format Specification. Revision 11 – June 20, 2017. Available at: https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/pecoff.docx.

Информация об авторах / Information about authors

Сергей Владимирович МИРОНОВ – кандидат физико-математических наук, доцент, декан факультета компьютерных наук и информационных технологий. Сфера научных интересов: методы сокращения диагностической информации с использованием словарей неисправностей, формальные языки и грамматики, функциональное программирование.

Sergei Vladimirovich MIRONOV – Candidate of Science in Physics and Mathematics, Associate Professor, Dean of the Faculty of Computer Science and Information Technologies. Research interests: methods of diagnostic information compression using fault dictionaries, formal languages and grammars, functional programming.

Инна Александровна БАТРАЕВА – кандидат физико-математических наук, доцент, заведующая кафедрой технологий программирования. Сфера научных интересов: дискретная математика, теория автоматов, теория формальных языков и грамматик, информационные системы в теоретической и прикладной лингвистике.

Inna Aleksandrovna BATRAEVA – Candidate of Science in Physics and Mathematics, Associate Professor, Head of the Department of Programming Technologies. Research interests: discrete mathematics, automata theory, theory of formal languages and grammars, information systems in theoretical and applied linguistics.

Павел Дмитриевич ДУНАЕВ – магистрант направления «Математическое обеспечение и администрирование информационных систем». Сфера научных интересов: компиляторы, операционные системы, дискретная математика.

Pavel Dmitrievich DUNAEV – Master's Student of Saratov State University. Research interests: compilers, operating systems, discrete mathematics.

DOI: 10.15514/ISPRAS-2022-34(5)-6

Natch: Определение поверхности атаки программ с помощью отслеживания помеченных данных и интроспекции виртуальных машин

[1,2] *П.М. Довгалюк, ORCID: 0000-0003-2483-5718 <pavel.dovgalyuk@ispras.ru>*
[1] *М.А. Климушенкова, ORCID: 0000-0001-6737-9092 <maria.klimushenkova@ispras.ru>*
[1] *Н.И. Фурсова, ORCID: 0000-0001-6817-4670 <natalia.fursova@ispras.ru>*
[1] *В.М. Степанов, ORCID: 0000-0003-2141-3333 <vladislav.stepanov@ispras.ru>*
[1] *И.А. Васильев, ORCID: 0000-0003-3824-2753 <ivan.vasiliev@ispras.ru>*
[1] *А.А. Иванов, ORCID: 0000-0003-2697-1449 <arkadiy.ivanov@ispras.ru>*
[1] *А.В. Иванов, ORCID: 0000-0001-9286-4719 <alexey.ivanov@ispras.ru>*
[1] *М.Г. Бакулин, ORCID: 0000-0002-8569-7382 <bakulinm@ispras.ru>*
[1] *Д.И. Егоров, ORCID: 0000-0001-8037-072X <egorov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Новгородский государственный университет им. Ярослава Мудрого, 173003, Великий Новгород, ул. Большая Санкт-Петербургская, д. 41*

Аннотация. Natch – это инструмент для получения поверхности атаки, то есть поиска исполняемых файлов, динамических библиотек, функций, отвечающих за обработку входных данных (файлов, сетевых пакетов) во время выполнения задачи. Функции из поверхности атаки могут быть причиной уязвимостей, поэтому им следует уделять повышенное внимание. В основе инструмента Natch лежат доработанные методы отслеживания помеченных данных и интроспекции виртуальных машин. Natch построен на базе полносистемного эмулятора QEMU, поэтому позволяет анализировать все компоненты системы, включая ядро ОС и драйверы. Собранные данные визуализируются в графическом интерфейсе SNatch, входящем в поставку инструмента. Построение поверхности атаки может быть встроено в CI/CD для интеграционного и системного тестирования. Уточненная поверхность атаки позволит поднять эффективность технологий функционального тестирования и фаззинга в жизненном цикле безопасного ПО.

Ключевые слова: динамический анализ; интроспекция; анализ помеченных данных; qemu; инструментирование; natch

Для цитирования: Довгалюк П.М., Климушенкова М.А., Фурсова Н.И., Степанов В.М., Васильев И.А., Иванов А.А., Иванов А.В., Бакулин М.Г., Егоров Д.И. Natch: Определение поверхности атаки программ с помощью отслеживания помеченных данных и интроспекции виртуальных машин. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 89-110. DOI: 10.15514/ISPRAS-2022-34(5)-6

Natch: using virtual machine introspection and taint analysis for detection attack surface of the software

[1,2] *P.M. Dovgalyuk, ORCID: 0000-0003-2483*-5718 *<pavel.dovgalyuk@ispras.ru>*
[1] *M.A. Klimushenkova, ORCID: 0000-0001-6737-9092 <maria.klimushenkova@ispras.ru>*
[1] *N.I. Fursova, ORCID: 0000-0001-6817-4670 <natalia.fursova@ispras.ru>*
[1] *V.M. Stepanov, ORCID: 0000-0003-2141-3333 <vladislav.stepanov@ispras.ru>*
[1] *I.A. Vasiliev, ORCID: 0000-0003-3824-2753 <ivan.vasiliev@ispras.ru>*
[1] *A.A. Ivanov, ORCID: 0000-0003-2697-1449 <arkadiy.ivanov@ispras.ru>*
[1] *A.V. Ivanov, ORCID: 0000-0001-9286-4719 <alexey.ivanov@ispras.ru>*
[1] *M.G. Bakulin, ORCID: 0000-0002-8569-7382 <bakulinm@ispras.ru>*
[1] *D.I. Egorov, ORCID: 0000-0001-8037-072X <egorov@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Yaroslav-the-Wise Novgorod State University, 41, B. Sankt-Peterburgskaya st., Novgorod, 173003, Russia*

Abstract. Natch is a tool that provides a convenient way of obtaining an attack surface. By attack surface we mean a list of executable files, dynamic libraries and functions that are responsible for input data processing (such as: files, network packets) during task execution. Functions that end up in the attack surface are possible sources of software vulnerabilities, so they should be given an increased attention during an analysis. At the heart of the Natch tool lay improved methods of tainted data tracking and virtual machines introspection. Natch is built on the basis of the full-system QEMU emulator, so it allows you to analyze any system components, including even the OS kernel and system drivers. The collected attack surface data is visualized by SNatch, which is tool for data post-processing and GUI implementation. SNatch comes with Natch tool by default. Attack surface obtaining can be built into CI/CD for integrational and system testing. A refined attack surface will increase the effectiveness of functional testing and fuzzing in the life cycle of secure software.

Keywords: dynamic analysis; introspection; taint analysis; qemu; instrumentation; natch

For citation: Dovgalyuk P.M., Klimushenkova M.A., Fursova N.I., Stepanov V.M., Vasiliev I.A., Ivanov A.A., Ivanov A.V., Bakulin M.G., Egorov D.I Natch: using virtual machine introspection and taint analysis for detection attack surface of the software. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 89-110 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-6

1. Введение

Один из видов поверхности атаки программного обеспечения (ПО) – это те функции, которые получают данные из недоверенных источников. В такие функции злоумышленник может подать некорректные данные и тем самым нарушить работу программы.

Разработчики определяют поверхность атаки своего ПО для тщательного тестирования (в том числе с помощью фаззинга) этих функций. Также поиск поверхности атаки необходим для того, чтобы найти зависимости от лишних или устаревших библиотечных функций. Кроме того, такой анализ может использоваться при сертификации программного обеспечения, чтобы убедиться, что оно достаточно хорошо протестировано.

При сертификации аналитики для нахождения поверхности атаки используют статический анализ (построение зависимостей функций, просмотр кода через IDE) или динамический анализ (поиск покрытия кода набором тестов). Статический анализ не способен в принципе находить некоторые типы зависимостей, к примеру, из-за динамического связывания.

Динамический анализ (при достаточно хорошем наборе тестов) находит именно тот код, который выполнялся. Но сведения о выполненных функциях (и строках кода) недостаточно точно отражают характер этих функций. Зависят ли они от входных данных? Например, код

инициализации выполняется всегда, но злоумышленник редко может на него повлиять. Поэтому этот код не должен быть включён в поверхность атаки.

Для определения более точной поверхности атаки необходимо отследить те программные модули, которые взаимодействовали со входными данными из недоверенных источников.

Мы предлагаем использовать для этого отслеживание помеченных данных в полносистемном эмуляторе [1]. Это позволяет проследить путь данных по всем исполняемым файлам и между процессами. Однако при использовании полносистемного эмулятора для анализа системы нет простого способа определить, что именно там выполняется. Это называется семантическим разрывом - код выполняется, но его высокоуровневая логика неизвестна. Для выделения из выполняющегося кода процессов, модулей, функций и других абстракций существуют методы интроспекции виртуальных машин [2].

В открытом доступе есть два инструмента для полносистемного динамического анализа помеченных данных для архитектуры x86-64: DECAF [3] и Panda [4]. Оба инструмента основаны на эмуляторе QEMU и используют динамическую бинарную трансляцию для отслеживания пометок. Однако ни в одном из этих инструментов нет достаточно развитых средств интроспекции для создания отчётов об искомой поверхности атаки.

В этой статье представлено развитие методов анализа помеченных данных и интроспекции, а также инструмент Natch, который использует их для нахождения поверхности атаки как отдельных приложений, так и целых систем из приложений, контейнеров и динамических библиотек.

2. Общая схема работы Natch

Natch предназначен для извлечения из хода работы системы подробностей о выполняемых приложениях: какие программные модули загружались, какие процессы были запущены, порядок вызова функций и т.п. Чтобы выделить из этого множества только те сущности, которые относятся к поверхности атаки (то есть обрабатывающие небезопасные данные), Natch помечает входные данные из недоверенных источников, а затем средствами эмулятора отслеживает их распространение по памяти виртуальной машины. Так можно узнать, где обрабатывались эти данные и получить поверхность атаки, интересующую аналитика.

Часть Natch, отвечающая за нахождение поверхности атаки, построена на основе эмулятора QEMU [5]. Эмулятор поддерживает множество аппаратных архитектур (в частности, x86, ARM, MIPS, RISC-V) и платформ на их основе.

Из возможностей по анализу системы в QEMU есть только эмулятор gdb-сервера для подключения отладчика и логирование выполняющихся инструкций. Для отслеживания потоков данных и определения того, какие именно приложения выполняются в гостевой системе, этого недостаточно.

Natch представляет собой набор плагинов [6] для Qemu, которые инструментируют выполняемый код и этим замедляют работу эмулятора. В связи с этим, анализ предлагается проводить с использованием детерминированного воспроизведения [7]. Это минимизирует воздействие средств анализа на изучаемую программу. При подключении инструментов анализа к виртуальной машине, воспроизводящей сценарий работы, поведение гостевой системы никак не изменится благодаря тому, что оно уже записано заранее. В нашем случае на записанный сценарий уже не будут оказывать влияния задержки от плагинов анализа, что очень важно для корректной работы часов реального времени и при взаимодействии с сетью.

Эмулятор QEMU поддерживает детерминированное воспроизведение для всех гостевых платформ, что позволяет использовать его как основу при создании средств полносистемного анализа и отладки.

Примерный алгоритм получения поверхности атаки может выглядеть следующим образом.

1) Запись тестового сценария средствами QEMU.

2) На этапе воспроизведения готового сценария выполняются следующие шаги:
   a) загрузка плагина natch;
   b) ожидание загрузки гостевой системы до интересующего аналитика момента работы системы;
   c) включение анализа помеченных данных (если он не был включен автоматически через конфигурационный файл);
   d) автоматический сбор информации о потоках данных в тестовом сценарии;
   e) сохранение собранных данных и завершение работы эмулятора.

3. Поиск поверхности атаки

При анализе кода, выполняющегося в виртуальной машине, возникает семантический разрыв между представлением программы и кода ОС как последовательности выполняющихся инструкций и тем представлением, которое необходимо для анализа (вызываемые функции, выполняющиеся процессы и т.п.). Для сокращения этого разрыва применяется интроспекция виртуальных машин (virtual machine introspection, VMI) [8, 9].

Современные подходы к интроспекции сопоставляют исходный код ядра ОС с выполняемыми инструкциями, чтобы получить сведения о расположении структур данных в памяти и их содержимом [10, 11] или встраивают свои модули в гостевую систему [4].

Первый подход привязывает алгоритмы анализа к конкретным операционным системам и даже их недокументированным внутренним структурам, если исходные коды ОС недоступны. Внедрение модулей анализа вторым подходом может менять работу системы, требует наличия SDK для сборки гостевого агента, а также не дает возможности использовать детерминированное воспроизведение работы системы.

Наш подход к интроспекции основан на перехвате редко меняющихся событий (системные вызовы) и анализе ядерных структур данных в памяти виртуальной машины.

Интерфейс системных вызовов меняется в рамках существующей системы очень редко [12]. Обычно только добавляются новые варианты функций или старые перестают использоваться.

Но данных, которые системные вызовы получают в виде параметров или возвращают как результат, недостаточно для восстановления подробной картины о работающих процессах и загруженных модулях. К примеру, невозможно связать системный вызов execve (куда передаётся имя программы) и порождённый им процесс.

Более подробные данные о системе извлекаются из ядерных структур данных. Структуры и адреса их расположения в памяти могут различаться в зависимости от версии и сборки ядра. В связи с этим, для применения такого подхода к каждой ОС необходим профиль интроспекции, который содержит смещения и адреса нужных структур и глобальных переменных. Построение такого профиля происходит во время отдельного запуска эмулятора с анализируемой системой.

3. Настройка интроспекции

Для построения профиля интроспекции применяются методы восстановления структур данных ядра. Такие методы можно разбить на 5 категорий: (1) с помощью компилятора [13, 14], (2) с помощью отладчика [15, 16], (3) с помощью гостевой программы [3, 4, 17, 18, 19], (4) с помощью бинарного анализа [20, 21, 22], (5) ручные подходы [23, 24]. Первые две категории предполагают наличие у пользователя исходного кода ядра или отладочных символов в сборке, что может быть доступно не для всех систем. Методы третьей категории требуют внесения изменений в гостевую ОС, что не всегда возможно в рамках выполняемой задачи анализа. Основанные на бинарном анализе методы сопоставляют бинарный код ядра с представлением системы, построенном на базе исходного кода или бинарного кода другой

версии ядра. Ручные подходы опираются на существенные усилия, которые разработчик прикладывает к разработке специальных алгоритмов восстановления для каждой структуры и поля.

Метод генерации профиля интроспекции, применяемый в нашей работе, следует отнести к ручным подходам. Мы применяем эвристические алгоритмы, которые сопоставляют параметры и возвращаемые значения системных вызовов с нужными для интроспекции полями структур данных ядра. Так как наборы системных вызовов в разных ОС похожи, то основные идеи эвристик, разработанных для одной системы, можно повторно использовать для восстановления структур данных ядра других ОС.

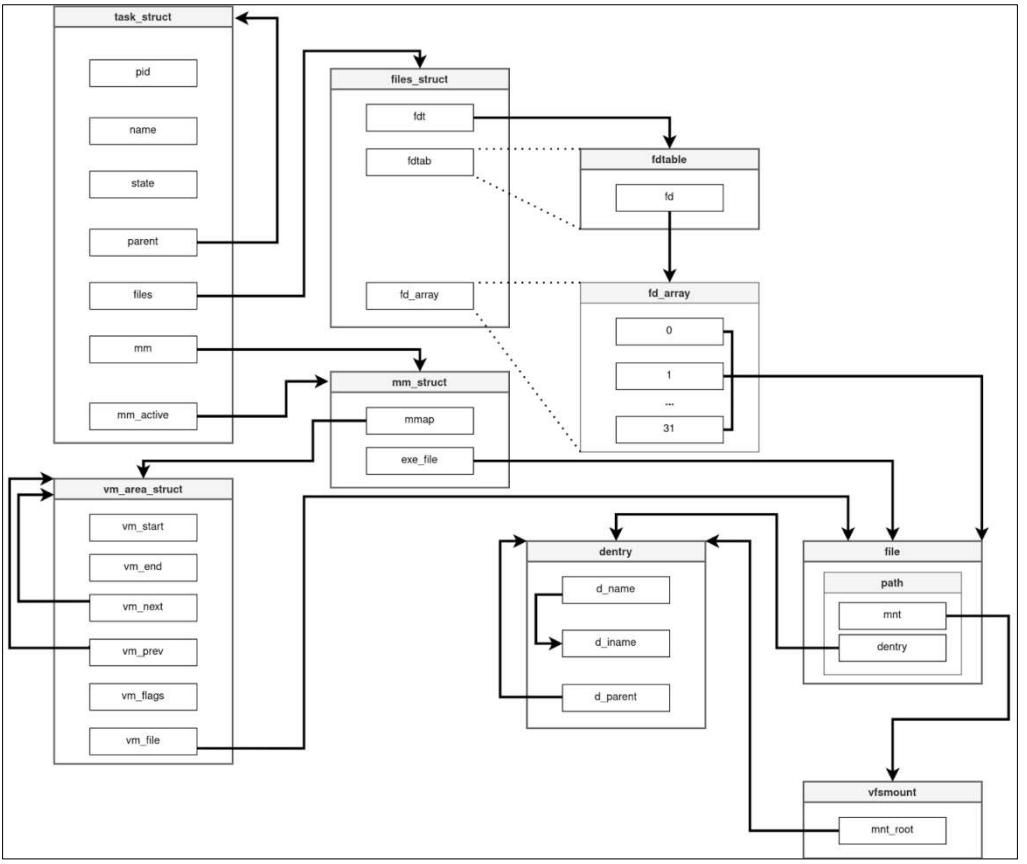


Рис. 1. Ключевые структуры данных ядра Linux
Fig. 1. Key Data Structures of the Linux Kernel

На рис. 1 представлены основные структуры и поля ОС Linux, которые восстанавливаются нашим подходом. Основная структура данных на этой схеме – это task_struct. Она описывает процесс.

На старых версиях Linux адрес task_struct текущего процесса можно извлечь в любой момент из стека ядра. Для платформы x86 адрес лежит на верхушке этого стека. Новые ядра Linux могут не хранить этот адрес в стеке. Вместо этого, адрес task_struct можно найти на определенном смещении от адреса из регистра GS. Определение этого смещения является первой задачей настройки интроспекции, так как в task_struct хранится много полезных атрибутов процесса (имя, загруженные модули, командная строка и т.п.). Чтобы найти task_struct, мы сначала перехватываем системный вызов getpid. В области 128 килобайт памяти от адреса GS происходит разыменование каждого возможного указателя. В каждой

возможной структуре task_struct ищется известное значение PID из системного вызова и строка, похожая на имя процесса (поле comm). Когда искомые значения повторно находятся на одних и тех же смещениях некоторое количество раз, смещения фиксируются в профиле интроспекции и используются в последующих фазах восстановления структур.

Поле parent в task_struct указывает на структуру, описывающую родительский процесс. Для определения смещения этого поля наше решение собирает значения адресов всех встречаемых task_struct. Затем ищет поле с указателем на другую такую структуру и проверяет, что путем разыменования этого поля можно дойти до корневого процесса, поле parent которого указывает на его же task_struct.

Поиск смещения поля state, которое описывает состояние процесса, опирается на тот факт, что в течение системного вызова exit в него записывается значение 40h или 80h (варианты константы TASK_DEAD для разных ядер). Поиск смещений файловых структур выполняется по завершению системного вызова open: нам нужны структуры dentry, поле name которых содержит имя файла, совпадающее с параметром системного вызова. Смещения указателей на пути к этим структурам перебираются в определенных диапазонах значений. На предполагаемых таблицах файловых дескрипторов используется значение файлового дескриптора из системного вызова для перехода к следующей структуре.

Смещения структуры mm_struct ищутся на момент завершения системных вызовов mmap. Поле exe_file содержит адрес структуры file и описывает исполняемый файл процесса. По ранее определенным смещениям файловых структур для каждого предполагаемого указателя на file, наше решение восстанавливает имя исполняемого файла и сопоставляет его с именем процесса. По известным смещениям также определяется адрес структуры file для файлового дескриптора из параметров системного вызова.

Поле mmap в mm_struct указывает на список структур vm_area_struct, которые описывают регионы памяти. Здесь мы применяем набор условий, которые для предполагаемого набора смещений полей должны выполняться одновременно. Поле vm_file первой структуры в списке должно указывать на ту же структуру file, что и exe_file. Поле vm_next должно указывать на следующую такую же структуру в списке. Для некоторых соседних структур в списке поле vm_start одной структуры должно совпадать с vm_end другой. Поле vm_file должно принимать некоторые специфичные значения. И последнее условие: в списке должна быть структура, описывающая новое отображение памяти с параметрами системного вызова mmap.

Также во время вызовов mmap определяются смещения полей arg_start, arg_end, env_start и env_end в структуре mm_struct, описывающих области памяти, где хранятся параметры командной строки и переменные окружения. Алгоритм следующий: выполняем обход списка структур vm_area_struct. В нем по значению поля vm_flags находим структуру, описывающую область памяти стека. Далее, находим указатели на четыре самых больших адреса в структуре mm_struct, которые принадлежат стеку. Это и будут искомые поля, потому что загрузчик в ядре Linux записывает искомые данные в самый конец стековой области памяти.

По завершению настроечного запуска, все найденные смещения структур записываются в конфигурационный файл. Этот файл затем используется для выполнения задач интроспекции.

4. Анализ помеченных данных

Динамический анализ помеченных данных [25] – метод, при котором отслеживается использование тех или иных данных в процессе выполнения программы. Выделяется три набора зависящих от задачи правил, определяющих анализ.

1) Какие данные отслеживать (когда вносить пометку); например, помечать данные из недоверенных источников, или помечать чувствительные данные (пароль)?
2) Как продвигать пометку: как операции над данными влияют на помеченность результата; например, при операциях копирования пометка также копируется, для бинарных операций результат будет помечен если был помечен хотя бы один исходный операнд?
3) Опционально: в каком случае выводить предупреждение или сообщать об ошибке? Например, если изначально помечались данные из недоверенных источников, попадание пометки в счётчик команд может свидетельствовать о попытке перехвата управления.

В открытом доступе есть два инструмента для полносистемного динамического анализа помеченных данных для архитектуры x86-64: DECAF [3] и Panda [4]. Оба инструмента основаны на эмуляторе QEMU и используют динамическую бинарную трансляцию для отслеживания пометок. Первый инструмент производит инструментирование на уровне внутреннего представления TCG, и благодаря различным оптимизациям является одним из самых быстрых известных на данный момент, однако базируется на старой версии эмулятора (1.0). Второй инструмент использует преобразование внутреннего представления TCG в LLVM-биткод, который инструментируется для продвижения пометок, благодаря чему поддерживается большое количество гостевых архитектур, но при этом значительно снижает скорость работы (более чем десятикратное замедление по сравнению с обычной эмуляцией).

В Natch используется метод «разбавленных» пометок [26], при котором чем больше преобразований происходило с помеченными данными, тем меньше «сила» полученной пометки. Это позволяет более точно описывать поверхность атаки, потому что злоумышленников в первую очередь интересуют функции, получающие слабо преобразованные данные. Если же входные данные подверглись, например, шифрованию, то сконструировать эксплоит будет слишком сложно.

Поэтому в отчётах Natch выводятся те функции, которые работали с помеченными данными с уровнем пометки не меньше заданного порога. Порог настраивается в конфигурационном файле и по умолчанию равен 250 (максимальное значение пометки равняется 255).

5. Монитор процессов

Плагин мониторинга процессов записывает историю созданий, переключений и завершений процессов. Для этого он выполняет чтение параметров task_struct из гостевой памяти. Создание нового процесса определяется по появлению нового экземпляра task_struct. Завершение процесса фиксируется в момент записи соответствующего значения в поле state этой структуры. Переключение текущего процесса проверяется, когда обновляется значение адреса каталога таблиц (регистр CR3).

Также плагин выполняет обход выполняемых процессов через поле parent, в ходе которого строит дерево процессов.

С помощью параметров mm_struct извлекается информация о регионах памяти процесса. По полю vm_flags в структуре vm_area_struct определяется принадлежность региона к разделяемой или приватной памяти. Также плагин извлекает адреса памяти со строками запуска процессов. За счет этих строк плагин определяет имена контейнеров docker.

6. Монитор файлов

Плагин мониторинга файлов отслеживает операции чтения и записи с файлами. Он может работать без чтения структур ядра. В таком случае он извлекает имена файлов из системных вызовов open. Эти имена привязываются к номерам файловых дескрипторов и процессам. Как правило, извлекаемые таким образом имена являются относительными. Чтобы получить полные имена файлов, плагин использует структуры ядра из гостевой памяти. По ним для файловых дескрипторов из системных вызовов он восстанавливает полные имена файлов.

Файлы сокетов в Linux не имеют уникальных имен. Их имена соответствуют типу сокетов, к которому они принадлежат. В нашей работе параметры сокетов, такие как порт и ip адрес, извлекаются из параметров системных вызовов bind и accept, затем сопоставляются адресам структур file.

Почему параметры сокетов сопоставляются структуре file, а не номерам файловых дескрипторов? Номера файловых дескрипторов идентифицируют файл в рамках одного процесса. Когда открытые файлы наследуются дочерним процессом после вызова fork, возникает необходимость привязывать сопоставленные параметры к номерам файловых дескрипторов нового процесса. В то же время, адрес структуры file остается одним и тем же в родительском и дочернем процессе, за счет чего применять его в качестве ключа для хранения параметров значительно проще.

7. Монитор модулей

Был разработан плагин, который определяет имена и адреса отображений в памяти для исполняемых модулей. Для этого он использует параметры системных вызовов mmap. Номер файлового дескриптора из аргументов mmap через монитор файлов преобразуется в название модуля.

Но из памяти виртуальной машины как правило невозможно прочитать секции исполняемых файлов с отладочной информацией, потому что операционной системе они не нужны. Поэтому основной набор исполняемых файлов, интересующих аналитика, загружается отдельно. Так Natch может прочитать из них всю отладочную информацию и сопоставить адреса функций с их именами.

Для обнаружения этих файлов в памяти, Natch сравнивает содержимое их кодовых секций с данными из страниц памяти в момент их отображения. Если загружаемая страница не относится ни к одному из этих файлов, плагин извлекает информацию об отображении памяти из структур ядра. Выполняется поиск структуры vm_area_struct, поля vm_start и vm_end которой соответствуют диапазону области памяти искомого модуля. Из поля vm_file данной структуры извлекается полное имя исполняемого файла.

8. Восстановление процессов и потоков

Для получения представления о поведении системы необходимо иметь данные о стеках вызовов, однако восстановление такой информации невозможно без заведомого получения данных о процессах и потоках в системе. Восстановление этих данных также должно следовать следующим принципам: работа в условиях полносистемного анализа, работа алгоритма без встраиваемых программ и программ-агентов, универсальность метода вне зависимости от анализируемой системы, возможность реализации для различных процессорных архитектур, алгоритм должен опираться на низкоуровневую информацию от системы. Таким образом полученный в результате алгоритм будет обеспечивать высокую переносимость, портируемость, простоту сопровождения и надежность результата.

Восстановление процессов является задачей тривиальной, которая решается в Natch схожим образом с существующими аналогами: для организации виртуальной памяти система выделяет для каждого процесса новое значение записи в таблице трансляции виртуальных адресов в физические. Процессор выделяет отдельный регистр для хранения этого значения (для x86 это CR3, для ARM – CP15 и т.п.). Таким образом, наблюдая за значением этого регистра, можно однозначно отличать друг от друга запущенные процессы.

Однако для восстановления информации о потоках обычного наблюдения за состоянием регистров недостаточно, требуется вводить дополнительную логику.

Системы, реализующие анализ уровня процессов, такие как Pin [27], Valgrind [28], DynamoRIO [29], как правило, представляют из себя виртуальные машины, внутри которых

запускается рассматриваемое приложение. Им свойственен прямой доступ к любой интересующей информации о рассматриваемом процессе – легкость доступа к данным и их полнота обеспечивается за счет выполнения инструментария анализа в одной операционной системе с исследуемым приложением, а значит и наличии доступа к системному API. Однако в этом случае невозможно анализировать межпроцессное взаимодействие, анализировать ядро ОС. При этом инструменты характеризуются строгой ОС зависимостью и не обеспечивают необходимой в вопросах безопасности изоляции инструмента и предмета анализа.

Среди систем, реализующих полносистемный анализ, можно выделить: Virtuoso [17], TEMU [30], DECAF [3, 31], PANDA [4], PyREBox [32], PEMU [33].

Virtuoso [17] применяет внедряемую программу-агент для обращения к интересующим данным. Трасса данного обращения анализируется и таким образом обнаруживаются адреса для получения данных о потоках из системных структур.

TEMU [30] реализует динамическое бинарное полносистемное инструментирование на базе эмулятора QEMU. Извлечение данных уровня операционной системы происходит с помощью специфического для каждой ОС семантического извлекателя. Причем для его функционирования на ОС семейства Windows используется загружаемый в систему модуль ядра

Создатели DECAF [3, 31] при разработке опирались на наработки TEMU и избавились от модификации ядра исследуемой системы, однако вся информация о процессах, потоках, модулях основывается на прямом обращении к структурам ядра исследуемой системы.

Процесс анализа в фреймворках PANDA [4] и PyREBox [32] базируется на ОС-специфичном конфигурационном файле, который в первом случае генерируется программой-агентом, а во втором заранее создается и распространяется вместе с фреймворком. PANDA также реализует и условно ОС-независимые способы восстановления данных, однако они дают результаты низкой точности: так, например, определение потоков по ASID дает ложные результаты, если одному процессу соответствует несколько потоков, а предлагаемый эвристический метод основан на определении смены потока по значительной смене указателя стека, что не всегда соответствует действительности.

Восстановление данных о потоках в PEMU [33] привязано к специфике процессора x86 и также имеет невысокую точность: поток идентифицируется по значению регистра esp с наложенной обнуляющей маской на нижние 12 бит.

Предлагаемый нами подход для восстановления данных о потоках строится на общих для различных ОС принципах, реализуется в условиях полносистемного динамического анализа, и позволяет получать результат без встраивания программ-агентов.

Суть предлагаемого метода заключается в следующем: каждому процессу соответствует один или более поток, при этом каждый поток однозначно принадлежит процессу. Идентификация процесса задача тривиальная и описана выше. Таким образом задача заключается в том, чтобы разделить в рамках процесса потоки друг от друга. Способ разделения строится на следующем наблюдении: каждый из потоков использует локальные переменные и оперирует адресами возврата, а значит должен поддерживать отдельный стек. Отсюда следует вывод, что для однозначной идентификации потока необходимо и достаточно рассматривать пару “идентификатор процесса” и “диапазон значений стека” (рисунок). Сумев выделить из общей памяти стека локальные стеки для потоков, мы сможем однозначно определить и сам поток (рис. 2).

Задача определения диапазона значений стека сводится к следующему: как понять, относится ли новое значение SP к тому же потоку, а значит свидетельствует о расширении диапазона значений стека, или сигнализирует о создании нового. Предлагаемый метод основывается на предположении, что в системе должен присутствовать определенный набор событий,

предшествующих операции создания нового стека и связанного с ним потока. Другие же события, связанные с изменением стека, служат для расширения границ диапазонов уже обнаруженных стеков.

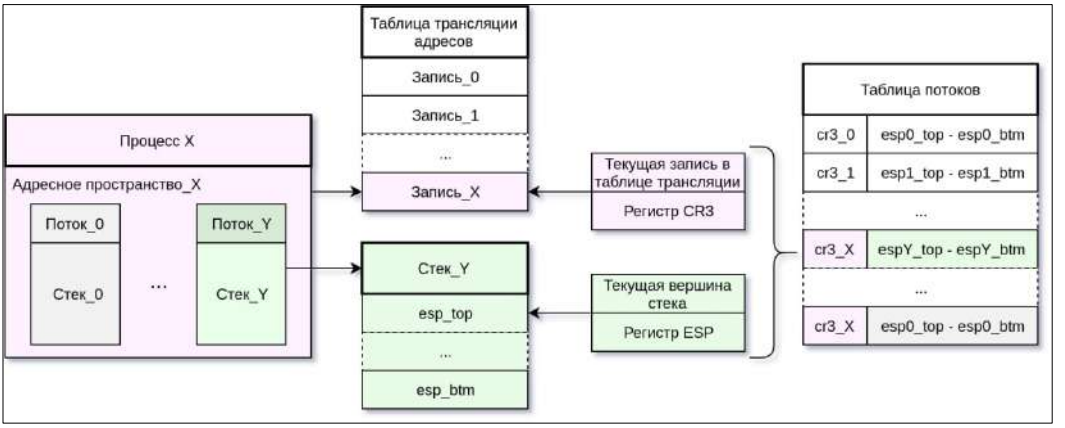


Рис. 2. Стеки потоков выполнения и адресные пространства процессов
Fig. 2. Control Flow Stacks and Process Address Spaces

В ходе разработки метода были выделены следующий события для создания новых и расширения существующих диапазонов: создание нового диапазона – возникновение нового значения SP после выполнения инструкции прямого присваивания SP нового значения, например: pop esp, mov esp; и возвращение из режима ядра в пользовательский режим. Прочие операции явно или косвенно модифицирующие стек расширяют диапазон. Данный набор событий будет отличаться для различных процессорных архитектур, однако принцип выделения подмножеств остаётся схожим.

9. Стеки вызовов

Имея данные о потоках, можно приступить к восстановлению стеков вызовов. При реализации данного алгоритма важно не опираться на соглашение о вызовах, т.к. информация об указателе фрейма не всегда описана в соглашении, а значит и восстановить стек вызова с его помощью в общем случае не удастся.

Логика работы алгоритма восстановления стека вызовов заключается в следующем: из выполняемых в гостевой машине инструкций выделяются те, что относятся к вызову функции (call), и те, что соответствуют операции возврата (ret). Для каждого из потоков сохраняется свой список адресов, по которым произошел вызов, и их расположение в стеке, а при возникновении возврата на требуемой глубине стека соответствующая запись из списка вызовов удаляется.

Однако для корректной реализации алгоритма важно было учитывать нестандартные способы системного взаимодействия со стеком, такие как: вызов процедуры с помощью безусловного или условного перехода; вызов процедуры через табличное значение; осуществление возврата через прямую модификацию стека и использование перехода; использование инструкции ret в качестве безусловного перехода; возврат через несколько записей в стеке вызовов; и ряд других. Только учитывая все особенности, нам удалось получить надежный алгоритм восстановления стеков вызовов. Данные, полученные от разработанного алгоритма, используются для последующего восстановления стека вызовов при обращении к помеченным данным, а также для построения графа вызовов, графа процессов, флейм-графа и пр., таким образом являются одной из базовых частей большой части реализуемых в Natch алгоритмов.

10. Покрытие кода

На данный момент существует множество инструментов, предназначенных для анализа покрытия. Они сильно различаются по целям и возможностям.

Компиляторы gcc и clang поддерживают сбор покрытия на уровне исходного кода. Сюда относится gcov [34] и его различные вариации. Главной проблемой такого рода инструментов является необходимость в исходном коде, который может быть недоступен. К преимуществам данного метода можно отнести поддержку большого числа архитектур и наличие инструментов для анализа результирующего покрытия в удобном формате, например, в HTML. (Сам формат gcov является достаточно неудобным для последующей обработки, о чём говорят и разработчики Lighthouse [35]). Кроме того, существуют решения, на основе данного подхода, позволяющие собирать покрытие бинарного кода для работающего ядра Linux [36]. Но и тут есть ограничения, инструменты такого рода не могут поддерживать файлы библиотеки динамической компоновки.

Другой подход к анализу покрытия – использование аппаратных механизмов трассировки. К настоящему моменту наиболее продвинутой реализацией такой технологии является Intel Processor Trace, возникшая вместе с пятым поколением процессоров от Intel. Трассировка вычислительного процесса позволяет получить полный поток исполнения некоторого приложения с минимальными издержками. Важным преимуществом технологии Intel PT является возможность отслеживать выполнение кода на максимально низком уровне – вплоть до выполнения отдельных инструкций. В отличие от сбора покрытия кода на уровне исходного кода, данная технология может работать как с исходным кодом, так и с бинарным. В качестве альтернативы инструментированию во время компиляции и случайному назначению идентификаторов базовых блоков, методы, основанные на IPT используют фактические адреса базовых блоков во время выполнения для отслеживания переходов между базовыми блоками.

Если же использовать данный метод для сбора покрытия без изменения, то возникает проблема с затратным декодированием инструкций во время анализа трассировки (более 85% процессорного времени) [37]. Так происходит из-за повторного анализа одних и тех же инструкций для одного и того же двоичного файла. Для решения данной проблемы уже существуют методы, которые с помощью кэширования инструкций увеличивают производительность вплодь до 40 раз.

Другой проблемой является неэффективное хранение данных для их последующей обработки: IPT может за несколько секунд сбрасывать гигабайты сжатых данных. Для избежания данной проблемы можно воспользоваться эвристическим обобщением данных на лету.

Наконец, главный недостаток – поддержка архитектур. Как уже говорилось выше, IPT - является особенностью процессоров Intel, начиная с пятого поколения, следовательно, сбор покрытия бинарного кода будет работать только для x86 архитектуры.

Технология динамического бинарного инструментирования (DBI) заключается во вставке в бинарный код анализирующих процедур, отвечающих за проведение необходимого анализа, модификацию и мониторинг исследуемой программы. Данные процедуры вызываются каждый раз, при достижении определенного участка кода или возникновения в программе определенного события (создание процесса, возникновения исключения и т.д.).

Данный подход не требует наличия исходного кода анализируемого приложения – работа происходит непосредственно с бинарными файлами. Число поддерживаемых архитектур в нём будет зависеть от реализованных в виртуальной машине архитектур.

Главная проблема данного подхода вытекает из-за его базирования на виртуальной машине - время работы. Помимо того, что выполнение кода в виртуальной машине сам по себе не быстрый процесс, так ещё и дополнительные временные расходы добавляются ко всей

программе, даже если пользователя интересует только отдельная ее часть, например, разделяемая библиотека.

Несмотря на недостатки, этот подход хорошо вписывается в инфраструктуру Natch. К тому же, его архитектуронезависимость позволяет в дальнейшем расширять его применение.

10.1 Сбор информации о покрытии

Рассмотрим три основных способа сбора информации о покрытии кода.

1) На основе выполненных инструкций. Данный способ позволяет инструментировать одну инструкцию за раз и является наиболее точным.
2) Сбор информации о покрытие кода на основе выполненных базовых блоков. Базовый блок (BB) – это последовательность инструкций с одним входом и одним выходом. Вместо одного анализируемого вызова для каждой инструкции, часто эффективнее вставить один анализируемый вызов для BB, тем самым сократив количество вызовов.
3) Сбор информации о покрытие кода на основе трассировки. Под трассировкой понимают последовательность базовых блоков, которая завершается безусловной инструкцией, изменяющей поток управления (например, jmp/call/ret). Имеет один вход и несколько выходов. Может улучшить производительность, по сравнению с BB и INS.

Мы остановились на втором варианте, так как он даст меньшее замедление, чем сбор покрытия на основе выполненных инструкций и избавит от необходимости реализовывать сложные абстракции для трассировки в QEMU.

10.2 Реализация сбора покрытия

Каждый раз, когда блок кода выполняется в эмуляторе, кроме самого диапазона адресов этого блока, в покрытие кода добавляется информация о модуле и процессе, к которым блок трансляции принадлежит. Происходит это благодаря реализованным плагинам [6], которые способны находить загруженные модули и процессы для них.

Помимо подписки на начало выполнения блока трансляции, необходимо также подписаться и на возникающие исключения, разделяющие блок трансляции на две части: до и после. Для этого добавим новый сигнал после вычисления PC инструкции, вызвавшей исключение, и подпишемся на него в инструменте. После получения такого сигнала, необходимо уменьшить размер блока трансляции, до требуемого. Откинутая же часть станет позже новым блоком.

Основное различие между сбором покрытия бинарного кода для ядра и для пользовательского режима заключается в размерности адреса базовых блоков. Самым очевидным решением является использовать большую размерность адреса для всех базовых блоков, но это сильно увеличит объём файла с покрытием. Поэтому было решено определять, какой код выполняется в режиме ядра и указывать для него смещение не от начала модуля, а от начала секции .text в нём.

Сбор покрытия бинарного кода для помеченных данных будет аналогичен сбору покрытия бинарного кода для всех выполненных базовых блоков, за исключением поиска тех самых помеченных данных. При чтении помеченных данных, последний полученный блок трансляции будет считаться работающим с помеченными данными, следовательно, будет добавлен к результирующему покрытию.

10.3 Анализ полученного покрытия бинарного кода

В данный момент используется метод, основанный на получении выполненных базовых блоков и объединения их с функциями в IDA Pro [38], с помощью плагина. В процессе работы с плагином пользователь может выбрать интересующие его процессы и модуль (если он не определился автоматически) и просмотреть выполненные и помеченные базовые блоки на

графе потока управления. В дальнейшем планируется интеграция этого плагина с инструментом Lighthouse [35].

10.4 Сравнение с существующими решениями

В сжатой форме результаты сравнения приведены в табл. 1.

Табл. 1. Сравнение разных инструментов для сбора покрытия кода.

Решение	Независимость от компилятора	Инструментирование на уровне ядра	Инструментирование за пределами виртуальной машины	Поддержка архитектур	Пометка данных	Информация о процессах и модулях	Удобный формат
kcov	X	✓	X	✓	X	X	✓/X
Llvm-cov	X	X	X	✓	X	X	✓/X
IPT	✓	✓	X	X	X	✓	X
DynamoRIO	✓	X	X	✓	X	✓/X	✓
Valgrind	✓	X	X	X	X	✓	✓
PIN	✓	X	X	✓	X	✓	✓
PEMU	✓	✓	✓	✓	X	✓	✓
Panda	✓	✓	✓	✓	✓	✓	X
Natch	✓	✓	✓	✓	✓	✓	✓

Cov %	Address	Instr. hit	Func Name
87.50	0x1000	7/8	.init_proc
100.00	0x1080	2/2	sub_1080
100.00	0x11c9	46/46	add_to_tree
100.00	0x10a0	2/2	sub_10A0
100.00	0x10c0	2/2	sub_10C0
92.31	0x10e0	12/13	_start
100.00	0x10d0	2/2	sub_10D0
92.86	0x1180	13/14	__do_global_dtors_aux
100.00	0x11c0	2/2	frame_dummy
71.43	0x1140	10/14	register_tm_clones
100.00	0x14d8	4/4	.term_proc
98.04	0x1359	50/51	main
100.00	0x1460	34/34	__libc_csu_init
100.00	0x1090	2/2	sub_1090
100.00	0x1269	37/37	tree_to_array
100.00	0x12eb	32/32	sort_tree
55.56	0x1110	5/9	deregister_tm_clones

Рис. 3. Окно выбора функции из плагина отображения покрытия к IDA Pro
Fig. 3. Function selection window from the coverage display plug-in for IDA Pro

10.5 Примеры работы

На рис. 3-4 приведены окно выбора функции из плагина к IDA Pro и граф базовых блоков с подсвеченными блоками из того же плагина.

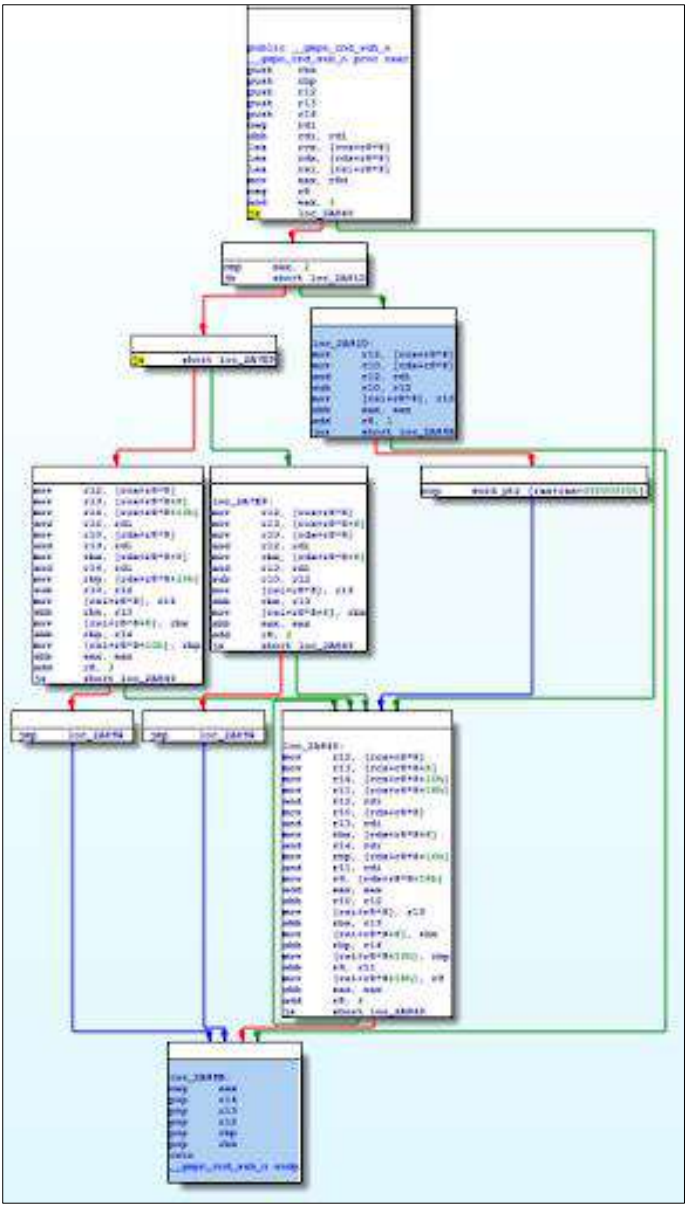


Рис. 4. Граф базовых блоков с подсвеченными блоками из плагина отображения покрытия к IDA Pro
Fig. 4. Graph of basic blocks with highlighted blocks from the coverage display plugin for IDA Pro

11. Графы взаимодействия процессов

Для анализа потоков данных между процессами строится граф их взаимодействия. Он иллюстрирует распространение помеченных данных по всей системе, показывая между какими процессами эти данные передавались. В графе комбинируется низкоуровневая

информация о распространении меток taint-анализа с высокоуровневой информацией о гостевой системе, получаемой в ходе интроспекции.

Близким аналогом предлагаемого решения является инструмент Panorama [39]. Это реализованная на базе эмулятора QEMU система обнаружения и анализа вредоносных программ в ОС Windows. Каждому помеченному байту Panorama сопоставляет небольшую структуру данных, хранящую источник данных и другую информацию, на базе которой строится граф. Взаимодействия отслеживаются между процессами, в которых произошла запись или пометка данных, и процессами, прочитавшими эти данные. Кроме процессов граф отображает модули анализируемой программы, файлы, сетевые соединения и другие источники ввода. Panorama распознает не только взаимодействия через ОЗУ, но и потоки данных через запись и чтение диска.

Инструменты расследования вторжений BackTracker [40], Taser [41], Protracer [42] и Rain [43] отслеживают информационные потоки между процессами в Linux за счет встраиваемых в ядро модулей. В отличие от Panorama, которая применяет полноценный динамический анализ байтовой гранулярности, вышеперечисленные инструменты опираются только на отслеживание системных вызовов, за счет чего имеют более грубый характер распространения меток и определения зависимостей.

В нашем решении для отслеживания передаваемых данных на эмуляторе выделяется дополнительный объем теневой памяти в размере двух байт на каждый байт ОЗУ гостевой системы. В эти два байта записывается идентификатор узла графа. Идентификаторам сопоставляются процессы, в течение работы которых записываются помеченные данные. Когда происходит чтение помеченных данных, проверяются идентификаторы читаемой памяти. Если они не соответствуют текущему процессу, то фиксируется передача данных от другого процесса.

Также в граф записывается информация о файлах и сокетах, в которые попадают помеченные данные. Взаимодействия с ними отслеживаются по системным вызовам. Имена файлов и параметры сокетов восстанавливаются соответствующими механизмами интроспекции.

Результат анализа записывается в json файл по завершению работы эмулятора. Этот файл затем используется инструментом SNatch для визуализации.

12. SNatch – интерфейс для визуализации результатов анализа

SNatch – это инструмент, предназначенный для постобработки, анализа и отображения данных, полученных от инструмента Natch. Состоит из двух частей: бэкенда и фронтенда. Первый выполняет разбор и обработку бинарных логов, сгенерированных Natch, и последующую генерацию сущностей для пользовательского анализа, на основании полученных данных. Фронтенд предоставляет графический пользовательский интерфейс, доступный через браузер, который позволяет изучать и взаимодействовать с полученными от бэкэнда данными для анализа.

Анализ организуется с помощью “проектов”, которые создаются пользователем через браузерный интерфейс. При создании нового проекта в бэкенд передаются сгенерированные Natch: лог помеченных данных, лог процессов, лог модулей, лог стеков вызовов, символьный лог и лог задач. Формируемые при разборе и анализе данные однозначно соотносятся с пользовательским проектом и используется для дальнейшей визуализации в фронтенд части.

В фронтенд части доступны следующие интерактивные модули отображения информации: граф вызовов, граф взаимодействия процессов, граф времени жизни процессов и дерево процессов.

Граф вызовов (рис. 5) традиционно представлен в виде древовидной структуры и содержит информацию о функциях, взаимодействовавших с помеченными данными. Интерактивные элементы позволяет сворачивать и разворачивать как отдельные ветви, так и дерево целиком.

Деревья разбиваются по процессам, а для каждой записи отображается смещение вызванной функции в модуле и название модуля, функции и номер строки в исходнике, если соответствующие данные были получены при создании проекта.

Рис. 5. Граф вызовов функций в SNatch
Fig. 5. Graph of function calls in SNatch

Граф взаимодействия процессов (рис. 6) представлен в виде ориентированного графа, где в вершинах расположены взаимодействующие элементы, а стрелки отображают направление и объем (толщина стрелки) передаваемых данных. На графе могут присутствовать вершины следующих типов: бинарный файл, интерпретатор и связанный с ним скрипт, сокет, внешняя сеть, терминал и текстовый файл. Благодаря шкале времени, можно поэтапно проследить процесс распространения помеченных данных внутри системы.

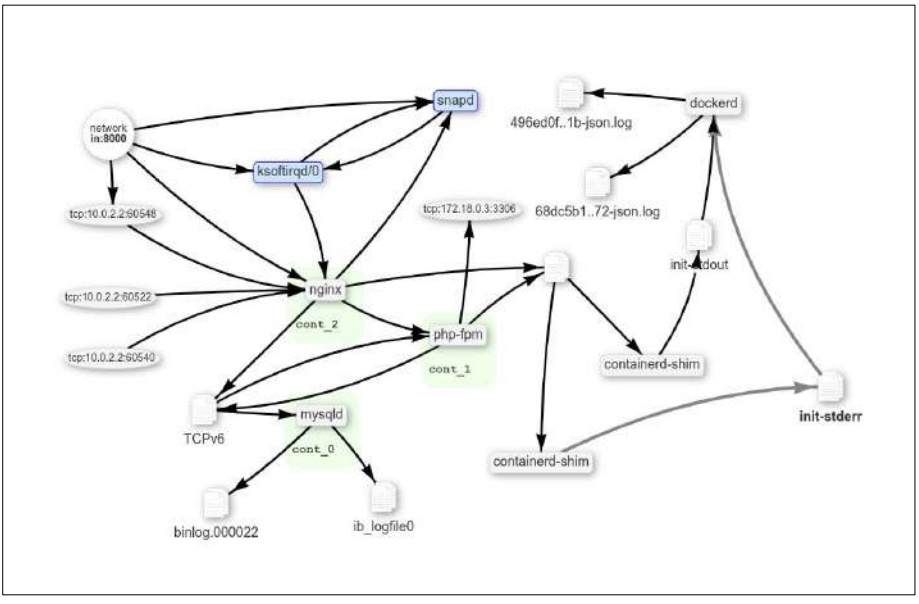


Рис. 6. Граф взаимодействия процессов в интерфейсе SNatch
Fig. 6. Process interaction graph in the SNatch interface

При этом отображение поддерживает несколько режимов в зависимости от привязки к моменту выполнения конкретного взаимодействия: active – активное взаимодействие на

выбранный момент времени; past – активное взаимодействие и произошедшие ранее; significant – активное взаимодействие и те вершины, которые еще будут использоваться на последующих шагах; и all – активное взаимодействие и все вершины, присутствующие в графе. Во всех режимах активные элементы имеют яркий цвет, прочие раскрашены в оттенки серого. Отдельно важно отметить поддержку отображения контейнеров — вершины, принадлежащие одному контейнеру, будут располагаться на плоскости рядом, выделены общим цветным блоком с именем контейнера. Интерактивные элементы данного графа позволяют как переключать описанные выше режимы, так и изменять время активного взаимодействия, а также свободно перемещать вершины графа по плоскости, приближать и отдалять граф.

Граф времени жизни процессов (рис. 7) представлен в виде временной шкалы, на которой изображены процессы в соответствии с временем и продолжительностью работы. Каждый из процессов расположен на отдельной строке графа, при этом принадлежащие одному контейнеру процессы располагаются по соседству и визуально выделены общим фоном. Граф поддерживает увеличение и уменьшение масштаба для поддержки анализа процессов короткой продолжительности, либо напротив, получения общего представления о возникавших в процессе работы системы процессах. При наведении мыши на процесс появляется всплывающий элемент, где отображается: имя процесса, идентификатор процесса, идентификаторы пользователей, путь к исполняемому файлу, строка запуска, родительский процесс и дочерние процессы. Оттенками красного на графе изображаются процессы, запущенные от root-пользователя, остальные - оттенками синего. Темными цветами выделяются процессы, которые оперировали помеченными данными. По двойному клику по таким процессам осуществляется переход на граф взаимодействия процессов в момент, когда выбранный процесс первый раз появляется на графе.

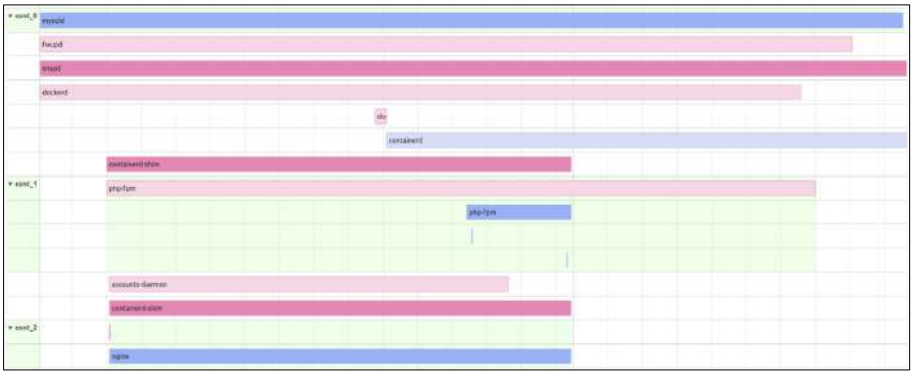


Рис. 7. Граф времени жизни процессов в интерфейсе SNatch
Fig. 7. Process lifetime graph in the SNatch interface

13. Примеры применения

Для демонстрации работоспособности Natch и SNatch рассмотрим несколько примеров.

Первый пример (рис. 8) продемонстрирует работу через пометку файлов. Сценарий следующий: есть две программы на языке C, одна из них читает файл (который и будет помечен), в первой строке которого хранится адрес сайта (в нашем случае www.google.ru), далее строка с этим адресом используется как параметр для вызова второй программы, которая запускает утилиту curl с переданным аргументом. Таким образом мы наблюдаем как помеченные данные передаются от процесса к процессу (в том числе и через командную строку) и в итоге утекают в сеть.

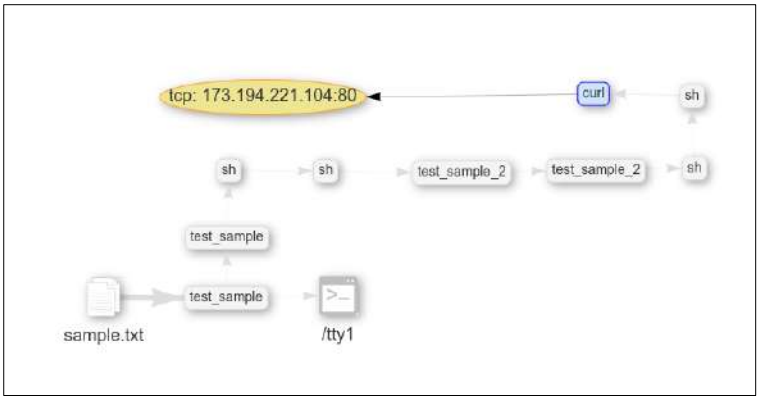


Рис. 8. Граф потоков данных между процессами в первом примере
Fig. 8. Graph of data flows between processes in the first example

Второй пример (рис. 9) тоже использует пометку файлов, но еще демонстрирует взаимодействие программ через сокеты. Сценарий примера: есть два Python скрипта – сервер и клиент. Для каждого подготовлен файл с сообщениями, которыми они будут обмениваться во время сессии (эти файлы помечаются). “Чат” клиента и сервера записывается в файл лог, который затем архивируется.

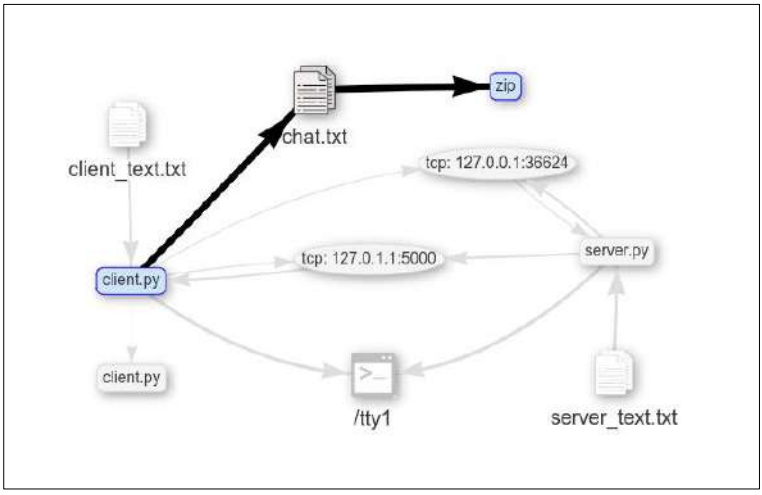


Рис. 9. Граф потоков данных между процессами во втором примере
Fig. 9. Graph of data flows between processes in the second example

14. Ограничения и планы развития

Текущая версия инструмента Natch способна работать для платформы x86_64 и ОС на основе ядер Linux и FreeBSD. Это ограничение возникло лишь из приоритетов развития, поэтому в дальнейшем будет добавлена поддержка процессорной архитектуры ARM и гостевой ОС Windows.

Другое ограничение – невозможность работы с вредоносным ПО – носит фундаментальный характер. Отслеживанию работы помеченных данных можно препятствовать, используя неявные каналы передачи информации, зависимости по управлению и т.п. Злоумышленник может создать программу, в которой поток данных отслеживаться не будет, поэтому Natch предназначен только для поиска поверхности атаки при разработке и сертификации, то есть когда разработчик не пытается противодействовать анализу поведения кода.

Извлекаемых методами интроспекции данных из виртуальной машины достаточно для построения более сложных и подробных отчётов, для навигации по исходному коду, для поиска обращений к заданным файлам. Поэтому графический инструмент SNatch планируется развивать для ещё большего упрощения работы аналитика.

Список литературы

[1] Климушенкова М.А., Бакулин М.Г. и др. О некоторых ограничениях полносистемного анализа помеченных данных. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 11-26 / Klimushenkova M.A., Bakulin M.G. et al. On Some Limitations of Information Flow Tracking in Full-system Emulators. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 11-26 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-1.

[2] Фурсова Н.И., Довгалюк П.М. и др. Легковесный метод интроспекции виртуальных машин. Программирование, том. 43, вып. 5, 2017 г., стр. 39-47 / Fursova N.I., Dovgalyuk P.M. et al. A lightweight method for virtual machine introspection. Programming and Computer Software, vol. 43, issue 5, 2017, pp. 307–313.

[3] Davanian A., Qi Z. et al. DECAF++: Elastic Whole-System Dynamic Taint Analysis. In Proc. of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID), 2019, pp. 31-45.

[4] Dolan-Gavitt B., Leek T. et al. Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection. In Proc. of the 20th ACM Conference on Computer and Communications Security (CCS), 2013, pp. 839-850.

[5] Qemu. A generic and open source machine emulator and virtualizer. URL: https://www.qemu.org/.

[6] Васильев И.А., Фурсова Н.И. и др. Модули для инструментирования исполняемого кода в симуляторе qemu. Проблемы информационной безопасности. Компьютерные системы, 2015 г., вып. 4, стр. 195-203 / Vasilev I., Fursova N. et al. Modules for instrumenting the executable code in QEMU simulator. Information Security Problems. Computer System, 2015, pp. 195-203 (in Russian).

[7] Dovgalyuk P. Deterministic Replay of System's Execution with Multi-target QEMU Simulator for Dynamic Analysis and Reverse Debugging. In Proc. of the 2012 16th European Conference on Software Maintenance and Reengineering, 2012, pp. 553-556.

[8] More A., Tapaswi S. Virtual machine introspection: towards bridging the semantic gap. Journal of Cloud Computing, vol. 3, 2014, article no. 16.

[9] Dovgalyuk P., Fursova N. et al. QEMU-based Framework for Non-Intrusive Virtual Machine Instrumentation and Introspection. In Proc. of the 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 944-948.

[10] Henderson A., Prakash A. et al. Make It Work, Make It Right, Make It Fast: Building a Platform-neutral Whole-system Dynamic Binary Analysis Platform. In Proc. of the 2014 International Symposium on Software Testing and Analysis, 2014, pp. 248-258.

[11] Hizver J., Chiueh T.-C. Real-Time Deep Virtual Machine Introspection and Its Applications. ACM SIGPLAN Notices vol. 49, issue 7, 2014, pp. 3-14

[12] Dovgalyuk P., Vasiliev I. et al. Non-intrusive Virtual Machine Analysis and Reverse Debugging with SWAT. In Proc. of the IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), 2020, pp. 196-203.

[13] Lin Z., Rhee J. et al. SigGraph: Brute Force Scanning of Kernel Data Structure Instances Using Graph-based Signatures. In Proc. of the Network and Distributed System Security Symposium (NDSS), 2011, 18 p.

[14] Schneider C., Pfoh J., Eckert C. Bridging the semantic gap through static code analysis. In Proc. of the 5th European Workshop on Systems Security (EuroSec), 2012, 6 p.

[15] Volatility 3: The volatile memory extraction framework. Available at: https://github.com/volatilityfoundation/volatility3, accessed 06.12.2022.

[16] LibVMI. Available at: https://libvmi.com/, accessed 06.12.2022.

[17] Fu Y., Lin Z. Space traveling across VM: Automatically bridging the semantic gap in virtual machine introspection via online kernel data redirection. In Proc. of the IEEE Symposium on Security and Privacy. 2012, pp. 586-600.

[18] Saberi A., Fu Y., Lin Z. Hybrid-bridge: Efficiently bridging the semantic gap in virtual machine introspection via decoupled execution and training memorization. In Proc. of the 21st Annual Network and Distributed System Security Symposium (NDSS'14), 2014, 15 p.

[19] Franzen F., Holl T. et al. Katana: Robust, Automated, Binary-Only Forensic Analysis of Linux Memory Snapshots. In Proc. of the 25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2022), 2022, 18 p.
[20] Zhang S., Meng X., Wang L. An adaptive approach for Linux memory analysis based on kernel code reconstruction. EURASIP Journal on Information Security, 2016, article no. 14, 13 p.
[21] Diaphora: A Free and Open Source Program Diffing Tool. Available at: http://diaphora.re/, accessed 06.12.2022.
[22] Feng Q., Prakash A. et al. Origen: Automatic extraction of offset-revealing instructions for crossversion memory analysis. In Proc. of the 11th ACM Asia Conference on Computer and Communications Security, 2016, pp. 11-22.
[23] Jiang X., Wang X. "Out-of-the-box" Monitoring of VM-based High-Interaction Honeypots. Lecture Notes in Computer Science, vol. 4637, 2007, pp. 198-218.
[24] Dolan-Gavitt B., Srivastava A. et al. Robust signatures for kernel data structures. In Proc. of the 16th ACM Conference on Computer and Communications Security, 2009, pp. 566-577.
[25] Schwartz E.J., Avgerinos T., Brumley D. All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). In Proc. of the IEEE Symposium on Security and Privacy, 2010, pp. 317-331.
[26] Bakulin M., Klimushenkova M., Egorov D. Dynamic Diluted Taint Analysis for Evaluating Detected Policy Violations. In Proc. of the 2017 Ivannikov ISPRAS Open Conference (ISPRAS), 2017, pp. 22-26.
[27] Luk C.-K., Cohn R. et al. Pin: building customized program analysis tools with dynamic instrumentation. ACM SIGPLAN Notices, vol. 40, issue 6, 2005, pp. 190-200.
[28] Nethercote N., Seward J. Valgrind: a framework for heavyweight dynamic binary instrumentation. In Proc. of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2007, pp. 89-100
[29] Bruening D., Duesterwald E., Amarasinghe S. Design and implementation of a dynamic optimization framework for Windows. In Proc. of the 4th ACM Workshop on Feedback-Directed and Dynamic Optimization (FDDO-4), 2001, 12 p.
[30] Song D., Brumley D. et al. BitBlaze: A new approach to computer security via binary analysis. Lecture Notes in Computer Science, vol. 5352, 2008, pp. 1-25.
[31] Henderson A., Lok Y. et al. DECAF: A Platform-Neutral Whole-System Dynamic Binary Analysis Platform. IEEE Transactions on Software Engineering, vol. 43, issue 2, 2017, pp. 164-184.
[32] Python scriptable Reverse Engineering Sandbox, a Virtual Machine instrumentation and inspection framework based on QEMU. Available at: https://github.com/Cisco-Talos/pyrebox, accessed 03.11.2022.
[33] Zeng J., Fu Y., Lin Z. PEMU: A Pin Highly Compatible Out-of-VM Dynamic Binary Instrumentation Framework. ACM SIGPLAN Notices, vol. 50, issue 7, 2015, pp 147-160.
[34] Stallman R.M. and the GCC Developer Community. Using the GNU Compiler Collection (GCC). Available at: https://gcc.gnu.org/onlinedocs/gcc-9.2.0/gcc.pdf, accessed 03.11.2022.
[35] Lighthouse - A Coverage Explorer for Reverse Engineers. Available at: https://github.com/gaasedelen/lighthouse/, accessed 03.11.2022
[36] Blasum H., Görgen F., Urban J. Gcov on an embedded system. In Proc. of the International Workshop on GCC for Research in Embedded and Parallel Systems (GREPS'07), 2007, 4 p.
[37] Husain A. Un-bee-lievable Performance: Fast Coverage-guided Fuzzing with Honeybee and Intel Processor Trace. Available at: https://blog.trailofbits.com/2021/03/19/un-bee-lievable-performance-fast-coverage-guided-fuzzing-with-honeybee-and-intel-processor-trace/, accessed 03.11.2022.
[38] IDA Pro, Available at: https://www.hexrays.com/ida-pro/, accessed 03.11.2022
[39] Yin H., Song D. et al. Panorama: capturing system-wide information flow for malware detection and analysis. In Proc. of the 14th ACM conference on Computer and communications security (CCS '07), 2007. Pp. 116-127.
[40] King S.T., Chen P.M. Backtracking intrusions. In Proc. of the 19th ACM Symposium on Operating Systems Principles, 2003, pp. 223–236.
[41] Goel A., Po K. et al. The taser intrusion recovery system. In Proc. of the 20th ACM Symposium on Operating Systems Principles, 2005, pp. 163-176.
[42] Ma S., Zhang X., Xu D. Protracer: Towards practical provenance tracing by alternating between logging and tainting. In Proc. of the Network and Distributed System Security Symposium, 2016, 15 p.

[43] Ji Y., Lee S. et al. RAIN: Refinable attack investigation with on-demand inter-process information flow tracking. In Proc. of the ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 377–390.

Информация об авторах / Information about authors

Павел Михайлович ДОВГАЛЮК – инженер, кандидат технических наук. Сфера научных интересов: интроспекция и инструментирование виртуальных машин, динамический анализ кода, отладчики, эмуляторы.

Pavel Mikhailovich DOVGALYUK – engineer, Ph.D. in Technical Sciences. Research interests: virtual machines introspection and instrumentation, dynamic analysis of code, debuggers, emulators.

Мария Анатольевна КЛИМУШЕНКОВА – разработчик программного обеспечения. Сфера научных интересов: отладка, интроспекция и инструментирование виртуальных машин, динамический анализ бинарного кода, эмуляторы.

Maria Anatolyevna KLIMUSHENKOVA is a software developer. Research interests: debugging, introspection and instrumentation of virtual machines, dynamic analysis of binary code, emulators.

Наталья Игоревна ФУРСОВА – инженер, кандидат технических наук. Сфера научных интересов: интроспекция и инструментирование виртуальных машин, динамический анализ кода, эмуляторы.

Natalia Igorevna FURSOVA – engineer, Ph.D. in Technical Sciences. Research interests: virtual machines introspection and instrumentation, dynamic analysis of code, emulators.

Владислав Михайлович СТЕПАНОВ – разработчик программного обеспечения. Сфера научных интересов: отладка, интроспекция и инструментирование виртуальных машин, динамический анализ бинарного кода, эмуляторы.

Vladislav Mikhailovich STEPANOV is a software developer. Research interests: debugging, introspection and instrumentation of virtual machines, dynamic analysis of binary code, emulators.

Иван Александрович ВАСИЛЬЕВ – разработчик программного обеспечения. Сфера научных интересов: отладка, интроспекция и инструментирование виртуальных машин, динамический анализ бинарного кода, эмуляторы.

Ivan Aleksandrovich VASILIEV is a software developer. Research interests: debugging, introspection and instrumentation of virtual machines, dynamic analysis of binary code, emulators.

Аркадий Алексеевич ИВАНОВ – разработчик программного обеспечения. Сфера научных интересов: отладка, интроспекция и инструментирование виртуальных машин, динамический анализ бинарного кода, эмуляторы.

Arkady Alekseevich IVANOV is a software developer. Research interests: debugging, introspection and instrumentation of virtual machines, dynamic analysis of binary code, emulators.

Алексей Владимирович ИВАНОВ – разработчик программного обеспечения. Сфера научных интересов: отладка, интроспекция и инструментирование виртуальных машин, динамический анализ бинарного кода, эмуляторы.

Alexey Vladimirovich IVANOV is a software developer. Research interests: debugging, introspection and instrumentation of virtual machines, dynamic analysis of binary code, emulators.

Максим Геннадьевич БАКУЛИН - разработчик программного обеспечения. Сфера научных интересов: эмуляция, динамический анализ помеченных данных, компьютерная безопасность.

Maksim Gennadievich BAKULIN is a sowtware engineer. Research interests: emulation, dynamic taint analysis, cyber security.

Данила Игоревич ЕГОРОВ - разработчик программного обеспечения. Сфера научных интересов: эмуляция, динамический анализ помеченных данных, компьютерная безопасность.

Danila Igorevich EGOROV is a sowtware engineer. Research interests: emulation, dynamic taint analysis, cyber security.

DOI: 10.15514/ISPRAS-2022-34(5)-7

Сравнение системы обнаружения вторжений на основе машинного обучения с сигнатурными средствами защиты информации

[1] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*
[2] *М.Н. Горюнов, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>*
[2] *А.Г. Мацкевич, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>*
[2] *Д.А. Рыболовлев, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Академия ФСО России,*
302015, Россия, г. Орел, ул. Приборостроительная, д. 35

Аннотация. В работе рассмотрен подход к сравнению систем обнаружения вторжений (СОВ) на основе нескольких независимых сценариев и комплексного тестирования, который позволил выявить основные достоинства и недостатки СОВ, основанной на применении методов машинного обучения (ML СОВ); определить условия, при которых ML СОВ способна превосходить сигнатурные системы по качеству обнаружения; оценить практическую применимость ML СОВ. Разработанные сценарии позволили смоделировать реализацию как известных атак, так и эксплуатацию уязвимости «нулевого дня». Сделан вывод о преимуществе ML СОВ при обнаружении ранее неизвестных атак, а также о целесообразности построения гибридных систем обнаружения, сочетающих возможности сигнатурного и эвристических методов анализа.

Ключевые слова: информационная безопасность; система обнаружения вторжений; машинное обучение; сигнатурные средства выявления атак; методика сравнения; сетевой трафик; компьютерная атака

Для цитирования: Гетьман А.И., Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Сравнение системы обнаружения вторжений на основе машинного обучения с сигнатурными средствами защиты информации. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 111-126. DOI: 10.15514/ISPRAS-2022-34(5)-7

A Comparison of a Machine Learning-Based Intrusion Detection System and Signature-Based Systems

[1] *A.I. Getman, ORCID: 0000-0002-6562-9008<ever@ispras.ru>*
[2] *M.N. Goryunov, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>*
[2] *A.G. Matskevich, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>*
[2] *D.A. Rybolovlev, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *The Academy of Federal Security Guard Service of the Russian Federation, 35, Priborostroitelnaya st., Oryol, 302015, Russia*

Abstract. The paper discusses the approach to the comparison of intrusion detection systems (IDS) that is based on several independent scenarios and comprehensive testing. This approach enabled to identify the advantages and disadvantages of the IDS based on machine learning methods (ML IDS), to identify the conditions under which ML IDS is able to outperform signature-based systems in terms of detection quality, to assess the practical applicability of ML IDS. The developed scenarios enabled to model the realization of both known attacks and a zero-day exploit. The conclusion is made about the advantage of ML IDS in the detection of previously unknown attacks and the feasibility of the construction of hybrid detection systems that combine the potential of signature-based and heuristic methods of analysis.

Keywords: information security; network intrusion detection system; machine learning; signature-based intrusion detection; comparison methodology; network traffic; computer attack

For citation: Getman A.I., Goryunov M.N., Matskevich A.G., Rybolovlev D.A. A Comparison of a Machine Learning-Based Intrusion Detection System and Signature-Based Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 111-126 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-7

1. Введение

Вопрос обеспечения безопасности информационной инфраструктуры является одним из наиболее острых и актуальных в настоящее время. Результаты исследований в этой области показывают постоянный рост количества и степени критичности реализуемых злоумышленниками атак [1]. Применяемые для их обнаружения средства защиты используют в основном сигнатурные методы анализа трафика, эффективность которых доказана, но ограничена полнотой базы решающих правил выявления известных информационных воздействий. Альтернативным подходом к обнаружению атак является применение эвристических анализаторов, построенных на применении технологий машинного обучения. Однако вопрос оценки их эффективности и практической применимости в общей системе защиты до сих пор остается малоисследованным.

Цель настоящей работы состоит в проведении сравнительного анализа систем обнаружения вторжений (СОВ), построенных на основе методов машинного обучения, и сигнатурных систем. Достижение сформулированной цели предполагает решение следующих задач:

- определение исходных условий для проведения сравнения;
- разработка сценариев и стендов для моделирования атак и фонового трафика;
- сравнение сигнатурных СОВ и ML СОВ;
- определение ограничений и рекомендаций по использованию эвристических анализаторов.

Новизна работы заключается в применении системного подхода к решению задачи сравнения качества обнаружения вторжений сигнатурных СОВ и ML СОВ.

2. Постановка задачи

Несмотря на достаточное количество работ, посвященных разработке моделей машинного обучения для систем обнаружения вторжений, в прямой постановке вопросы их эффективности в сравнении с сигнатурными средствами защиты в публикациях практически не обсуждаются.

В работе [2] рассматривается подход к тестированию эффективности межсетевых экранов веб-приложений (Web application firewall, WAF) в отношении SQL-инъекций (SQLi). Авторы рассматривают известные стратегии обнаружения SQLi уязвимостей, включая white-box и black-box тестирование, тестирование на основе модели, статический анализ кода, отмечают их недостатки, ограничивающие практическое применение. В исследовании предлагается новый вариант black-box тестирования на основе применения методов машинного обучения, позволяющий генерировать сценарии обхода защиты WAF. Сравнение с сигнатурными WAF выполнено на примере одного коммерческого решения и одного open-source решения – ModSecurity. Авторы отмечают обнаруженные сценарии обхода ModSecurity, однако при этом не приводятся конкретные примеры вредоносных запросов и не представлен анализ настроек средства защиты (версии средства и набора правил, paranoia level, anomaly score threshold и др.).

В статье [3] рассматривается вопрос повышения качества работы WAF ModSecurity за счет включения дополнительного модуля обнаружения аномалий на основе машинного обучения. В случае, когда данные обучения недоступны, предлагается использовать модель одноклассовой классификации (one-class classification). При наличии размеченной обучающей выборки отмечается возможность повысить качество обнаружения за счет применения модели машинного обучения с учителем – модели n-грамм (n-grams). При проведении практических экспериментов в исследовании используется одна из самых распространенных конфигураций WAF: WAF ModSecurity с набором правил OWASP CRS. Отмечается большое количество ложных срабатываний (до 40%) при использовании OWASP CRS, сложность и трудозатратность настройки и корректировки правил.

Для оценки качества предложенных решений используются три набора данных: общедоступные CSIC 2010 и ECML/PKDD 2007, а также авторский набор DRUPAL (трафик размечен при помощи WAF ModSecurity). Авторы отдельно подчеркивают важность задачи разработки публичных, общедоступных наборов данных, содержащих размеченный трафик с признаками актуальных компьютерных атак. Результаты исследования демонстрируют возможность повышения качества обнаружения атак WAF ModSecurity за счет включения модуля обнаружения аномалий. Вместе с тем полученные результаты свидетельствуют о возможности улучшения лишь метрики FP (false positive, количество ложных срабатываний) и не дают ответа на вопрос, как увеличить количество истинных срабатываний модели.

Работа [4] представляет собой обзор современных взглядов на построение WAF на основе методов машинного обучения. Отмечается, что при использовании сигнатурных WAF необходимо постоянно обновлять базы правил для поддержания возможности обнаруживать современные атаки. Кроме того, сигнатурные WAF практически не способны обнаруживать атаки «нулевого дня». Авторы подчеркивают эффективность применения моделей машинного обучения для противодействия ранее неизвестным атакам и простоту поддержания их в актуальном состоянии. Однако в статье не обсуждаются вопросы сравнения качества обнаружения сигнатурных и эвристических анализаторов.

В исследовании [5] подчеркивается низкая эффективность сигнатурных анализаторов в отношении обнаружения новых атак. В работе предложен новый метод обнаружения атак на основе применения sequence-to-sequence нейронных сетей. Метод прогнозирует ответ web-приложения и сравнивает, насколько прогноз отличается от реального ответа. Особенность предложенного подхода состоит в учете совокупности запросов и ответов веб-сервера (возможно, длительных во времени). По всем выбранным метрикам, кроме specificity,

предложенная модель продемонстрировала более высокое качество обнаружения атак в сравнении с WAF ModSecurity и установленным набором правил CRS. Вместе с тем авторы не уточняют настройки базы решающих правил (paranoia level, anomaly score threshold и др.), используемой в эксперименте. Кроме того, результаты оценивались на публичных наборах данных CSIC 2010 и CICIDS 2017, содержащих только известные атаки.

Настоящая работа является логическим продолжением исследования [6], в котором по результатам апробации синтезированной модели обнаружения атак, обученной и протестированной на реальных данных, показаны ее высокие показатели качества работы. Однако для более полного понимания преимуществ и недостатков разработанной модели обнаружения вторжений необходимо иметь данные в сравнении с результатами работы традиционных средств защиты в схожих условиях.

Решаемая в данном исследовании основная задача заключается в разработке практического подхода к сравнительному анализу сигнатурных СОВ и ML СОВ. Полученные результаты сравнения позволят сделать вывод о преимуществах и недостатках той или иной СОВ, а также сформулировать условия их практического применения с целью максимизации качества обнаружения атак.

3. Исходные данные

В представленном исследовании класс рассматриваемых компьютерных атак был ограничен веб-атаками, в частности: XSS, CSRF, SQL Injection, Bruteforce, Shell code, Command Injection.

Данное обстоятельство обусловлено параметрами разработанной модели машинного обучения для обнаружения компьютерных атак [6], а также связано с широким распространением этого класса атак, доступностью инструментов генерации, относительной простотой реализации тестовой инфраструктуры.

При выборе объекта защиты рассматривались 2 варианта:

- реальный объект в сети;
- «искусственный» объект в виде «чертовски уязвимого веб-приложения» DVWA (Damn Vulnerable Web Application) [7], развернутого в облачной среде с доступом в Интернет.

В табл. 1. сформулированы основные требования к объекту защиты, которые позволяют произвести полноценное моделирование и оценку качества работы систем обнаружения атак. В этой же таблице отражены оценки соответствия рассматриваемых объектов защиты данным требованиям.

Табл. 1. Оценка соответствия объектов защиты предъявляемым требованиям
Table 1. Assessment of compliance with the requirements for the objects of protection

№	Требование к объекту защиты	Реальный объект	DVWA
1	Возможность атаковать в реальном времени.	-	+
2	Наличие известных уязвимостей и средств их эксплуатации для сбора обучающего трафика.	+/-	+
3	Возможность доступа к объекту защиты для подключения, сбора и анализа данных.	+/- (затруднено)	+

Как видно из табл. 1, «искусственный» объект в виде веб-приложения DVWA, содержащего известные уязвимости, наиболее полно удовлетворяет всем требованиям. При этом результаты эксперимента на «искусственном» объекте защиты могут быть распространены на более общие практические случаи. В представленном исследовании за основу объекта защиты было выбрано веб-приложение DVWA.

При выборе сигнатурных средств защиты от веб-атак для сравнения были рассмотрены представители следующих двух классов средств защиты информации:

- межсетевые экраны веб-приложений (WAF);
- сетевые системы обнаружения атак (NIDS).

В настоящее время одним из наиболее распространенных межсетевых экранов веб-приложений с открытым исходным кодом является WAF ModSecurity [8]. Обычно он применяется с набором сигнатур обнаружения атак OWASP ModSecurity Core Rule Set (CRS) [9].

Основная задача CRS – обеспечить защиту веб-приложений от широкого перечня атак, включая атаки из OWASP Top Ten, с минимумом ложных срабатываний. В частности, обеспечивается защита от атак вида: SQL Injection, Cross Site Scripting, Local File Inclusion и др. Баланс между уровнями FN (False Negative) и FP (False Positive) данного средства обеспечивается настройкой PL (Paranoia Level) – от 1 до 5, при уровне 1 обеспечивается минимальное количество ложных срабатываний. При проведении экспериментов в исследовании использовался WAF ModSecurity с базой сигнатур CRS 3.3.2 от 30 июня 2021 г.

В качестве сетевой системы обнаружения атак в исследовании рассматривалась система обнаружения компьютерных атак уровня сети с открытым исходным кодом Suricata 6.0.3 [10] с базой решающих правил Proofpoint Emerging Threats Rules от 30 декабря 2021 [11]. База решающих правил использовалась «как есть», без дополнительных правок.

Эвристический анализатор была построен на основе синтезированной авторами модели типа «случайный лес» (ML COB) [6]. Обучение модели производилось на сбалансированной и предварительно обработанной выборке веб-атак, сформированной в рамках настоящего исследования (см. раздел 4.4). При формировании признакового пространства использовались признаки публичного датасета CICIDS2017. Тестирование ML COB и сигнатурных средств защиты осуществлялось на том же испытательном стенде, на котором были собраны обучающие данные, т.е. скоростные характеристики канала связи оставались неизменными.

Авторами было принято решение не рассматривать работу по незащищенному протоколу HTTP, поскольку сегодня его доля трафика составляет около 10%, а 90% – это трафик по защищенному протоколу HTTPS [12].

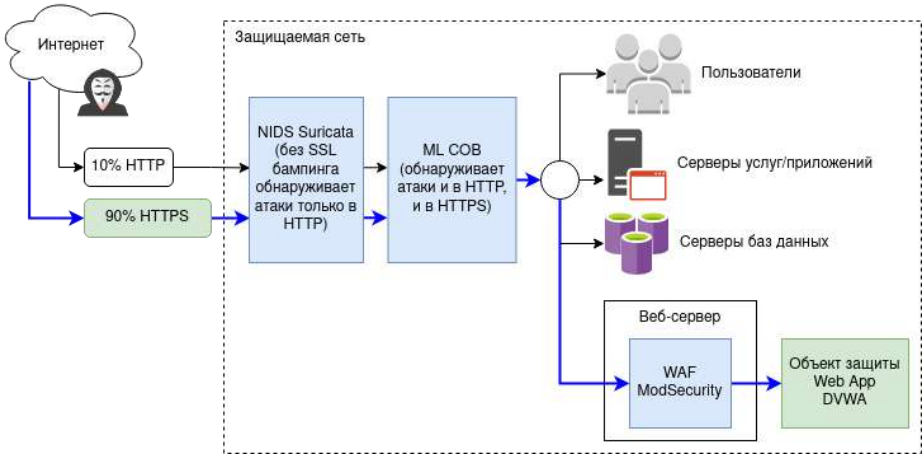


Рис. 1. Схема экспериментального стенда
Fig. 1. Scheme of the experimental stand

На рис. 1 представлена схема экспериментального стенда для сравнения рассматриваемых средств защиты. В левом верхнем углу расположен злоумышленник, который через сеть Интернет атакует объект защиты. Объект защиты – DVWA – расположен в правом нижнем углу. Стрелкой синего цвета показан путь прохождения вредоносных запросов от злоумышленника к объекту защиты. Ближе всего к объекту защиты расположен межсетевой экран WAF ModSecurity, на входе в защищаемую сеть в первом эшелоне средств защиты

установлена сетевая СОВ Suricata, после неё – разработанная авторами СОВ на основе машинного обучения.

В ходе проводимых экспериментов генерация атак осуществлялась по протоколу HTTPS, т.е. на уровне сети трафик являлся шифрованным. Как и предполагалось, NIDS Suricata без реализации механизма SSL бампинга не выявила ни одной атаки. В этой связи комплексное тестирование данного средства дополнительно осуществлялось на HTTP трафике. Для WAF ModSecurity, работающего на прикладном уровне, весь поступающий на анализ трафик является открытым, т.е. качество работы данного средства не зависит от использования защищенного протокола транспортного уровня. В отличие от рассмотренных сигнатурных средств ML СОВ показала способность обнаруживать атаки в зашифрованном трафике HTTPS на основе анализа признаков сетевых сессий без необходимости просмотра полезной нагрузки.

4. Методика сравнения разработанной модели с существующими средствами защиты

Получение объективной сравнительной оценки средств защиты может быть обеспечено при оценке качества их работы на нескольких независимых сценариях реализации атак и комплексном тестировании. В ходе проводимого исследования были разработаны сценарии, которые позволили смоделировать реализацию как известных атак, так и эксплуатацию 0-day уязвимости. Ниже приведено краткое описание предлагаемых сценариев сравнения.

Сценарий 1 – Внедрение и эксплуатация Shell-кода. Осуществляются попытки загрузки и эксплуатации Shell-кода «в обход» средств защиты с актуальными правилами обнаружения атак.

Сценарий 2 – Получение несанкционированного доступа к ресурсам системы на основе подбора или скрытого изменения пароля пользователя. Осуществляются попытки подбора пароля методом Brute Force, а также его несанкционированного изменения путем реализации атаки CSRF «в обход» средств защиты с актуальными правилами обнаружения атак.

Сценарий 3 – Моделирование эксплуатации 0-day уязвимости. Осуществляется реализация атаки типа SQL Injection «в обход» средств защиты с неактуальными правилами обнаружения атак: осуществляется искусственный «откат назад» (downgrade) базы решающих правил до момента времени, когда одна из реализаций атак стала 0-day уязвимостью.

Сценарий 4 – Комплексное тестирование. Осуществляется генерация различных типов атак и фонового трафика с использованием средств автоматизации, фиксация результатов и расчет основных показателей качества работы средств защиты.

Реакции сравниваемых средств защиты на разных стадиях реализации атак сценариев 1-3 оценивались только для HTTPS трафика. WAF ModSecurity был настроен на работу на первом уровне паранойи (Paranoia Level = 1).

4.1 Сценарий 1 – Внедрение и эксплуатация Shell-кода

Атака внедрения и эксплуатации Shell-кода состоит из двух стадий. На первой из них необходимо загрузить вредоносный код, а на второй – подключиться к нему. Первая стадия может быть реализована путем использования встроенных в веб-приложение механизмов загрузки, например, в DVWA имеется такой функционал. При этом для возможности последующей эксплуатации данный Shell-код должен корректно обрабатываться интерпретатором, т.е. для рассматриваемого случая загружаемый вредоносный файл должен иметь расширение «php» (ввиду того, что DVWA – это веб-приложение, написанное на языке программирования PHP).

Для формирования Shell-кода и последующего подключения к нему может быть использовано специализированное средство weevely [13]. Генерация Shell-кода с его помощью осуществляется следующей командой:

```
weevely generate password BackDoor.php,
```

где password – пароль для доступа к shell-коду;

BackDoor.php – имя генерируемого файла, содержащего shell-код.

Содержимое сгенерированного файла BackDoor.php обфусцировано и имеет следующий вид:

```
<?php
$q='$]Jo.=]J$t{$i}^$k{$j]J};}}]Jre]Jtur]J]Jn$o;]J}if(@preg_match("/]J$kh
(.+]J)$kf/]J",@file_]Jget]J_con]Jtents("ph]Jp://]J';
$u=str_replace('wj','','crewjwjatwjwje_wjfuwjnction');
$v='input"),$]J]Jm)==1){@ob_st]Jart(]J)]J;@e]Jval(@]Jgzuncompr]
Jess(@x(@b]Jase64_]Jdecode(]J$m[]J1])]J,$k)));$o=@]J]Job_ge';
$F='$k="5f4dc]J]Jc3b";$]Jkh="5a]Ja765d]J61d]J83";$]Jkf="27deb882cf99"]J;
$p=]J"hWwHxD]JH]JsxJ6]JEigxM";fu]Jnct]Jion x($t,$k)';
$P='{]J$c]J=str]Jlen($k)]J;$l=strlen($]Jt);$o=]J"]J";
for($i=0;]J$i<]J$]Jl;){for($j=0;(]J$j<]J$c&&$i<$]Jl);$j++]J,$i]J++){';
$D='t_c]Jontents();@o]J]J]J]Jb_]Jend_clean();]J$r=@base6]
J4_encode(@x(@g]Jzcompres]Js($o]J),$k));prin]Jt("$p]J$kh$r$kf");}';
$e=str_replace(']J','',$F.$P.$q.$v.$D);
$y=$u('',$e);$y();
?>
```

Загрузка файлов на сервер выполняется через штатные возможности DVWA на странице http://dvwa.isp/vulnerabilities/upload/. Для последующего подключения к загруженному Shell-коду используется следующая команда:

```
weevely http://dvwa.isp/hackable/uploads/BackDoor.php password.
```

В случае блокирования попыток прямой загрузки зловредного файла (срабатывания сигнатуры на расширение файла «php»), загрузка может быть осуществлена с использованием обходного варианта. Его суть заключается в следующем. Первоначально происходит загрузка файла без указания требуемого расширения, например, BackDoor.ph, а дальше осуществляется его переименование через реализацию атаки Command Injection. Для этого на странице http://dvwa.isp/vulnerabilities/exec/ необходимо отправить методом POST следующие данные:

```
1; cd /var/www/html/hackable/uploads; /???/?v Backdoor.ph Backdoor.php,
```

где /???/?v – маскировка команды /bin/mv.

В табл.2 представлены результаты реакции сравниваемых средств защиты на основных стадиях реализации атаки внедрения и эксплуатации Shell-кода.

Табл. 2. Реакция сравниваемых средств защиты на основных стадиях реализации атаки внедрения и эксплуатации Shell-кода

Table 2. Response of compared security systems at the main stages of the realization of the Shell code injection and exploitation attack

Средство защиты	**Стадия атаки и результат ее обнаружения**				**Примечания**
	Попытка загрузки Shell-кода (Backdoor.php)	**Попытка загрузки Shell-кода (Backdoor.ph)**	**Инъекция команды операционной**	**Подключение к Shell-коду**	
NIDS Suricata	–	–	–	–	Атаки в зашифрованном трафике HTTPS не обнаруживаются.

WAF ModSecurity	+	–	–	+	Попытка прямой загрузки Shell-кода (файл Backdoor.php) обнаруживается ввиду наличия в базе сигнатуры на соответствующее расширение файла. Попытка опосредованной загрузки Shell-кода (файл Backdoor.ph) не обнаруживается ввиду отсутствия в базе сигнатуры для указанного расширения файла. Реализация атаки Command Injection в части переименования файла с Shell-кодом не обнаруживается ввиду отсутствия соответствующей сигнатуры в базе. Подключение к Shell-коду обнаруживается.
ML COB	–	–	+	+	Попытка прямой и опосредованной загрузки Shell-кода не обнаруживается ввиду того, что обучение на таком типе трафика не проводилось; Реализация атаки Command Injection в части переименования файла с Shell-кодом обнаруживается. Подключение к Shell-коду обнаруживается.

Проведенный эксперимент показал следующее:

- NIDS Suricata не обнаруживает атаки сценария 1 в зашифрованном трафике HTTPS;
- WAF ModSecurity и ML COB практически сравнимы по результативности обнаружения атак внедрения и эксплуатации Shell-кода;
- качество обнаружения WAF ModSecurity зависит от полноты базы решающих правил, а ML COB – от качества обучающего трафика;
- в случае реализации опосредованной загрузки Shell-кода ML COB способна выявлять атаку на более ранней стадии (в частности, на стадии попытки внедрения команды операционной системы).

4.2 Сценарий 2 – Получение несанкционированного доступа к ресурсам системы на основе подбора или скрытого изменения пароля пользователя

Получение несанкционированного доступа к ресурсам системы может быть осуществлено двумя способами. Первый способ состоит в реализации атаки подбора пароля методом Brute Force, например, с помощью программного средства patator [14]. Второй способ заключается в несанкционированном изменении легального пароля пользователя на основе реализации атаки CSRF. Данная атака может быть осуществлена при наличии соответствующей уязвимости веб-приложения и переходе авторизованного пользователя на сайт злоумышленника, на котором от его имени скрыто выполняется запрос на смену пароля. Упомянутая уязвимость определяется отсутствием или некорректной реализацией механизма использования CSRF-токенов в целевом веб-приложении. Также для успешной реализации данной атаки должна быть деактивирована политика ограничения домена (Same Origin Policy) в браузере пользователя. Рассматриваемый эксперимент осуществлялся в предположении выполнения всех обозначенных условий.

В табл. 3 представлены результаты реакции сравниваемых средств защиты на реализацию атак подбора и несанкционированного изменения пароля пользователя.

Табл. 3. Реакция сравниваемых средств защиты на реализацию атак подбора и несанкционированного изменения пароля пользователя
Table 3. Response of compared security systems to the realization of Brute Force and CSRF attacks

Средство защиты	Тип атаки и результат обнаружения		Примечания
	Brute Force	CSRF	
NIDS Suricata	–	–	Атаки в зашифрованном трафике HTTPS не обнаруживаются.
WAF ModSecurity	–	–	Реализация атаки подбора пароля методом Brute Force не обнаруживается (правило для обнаружения таких атак отсутствует в базе правил). Реализация несанкционированной смены пароля за счет CSRF атаки не обнаруживается (правило для обнаружения таких атак отсутствует в базе правил).
ML COB	+	+	Реализация атаки подбора пароля методом Brute Force обнаруживается. Реализация несанкционированной смены пароля за счет CSRF атаки обнаруживается.

Проведенный эксперимент показал следующее:

- NIDS Suricata не обнаруживает атаки сценария 2 в зашифрованном трафике HTTPS;
- качество обнаружения WAF ModSecurity зависит от полноты базы решающих правил, а ML COB – от качества обучающего трафика;
- ML COB превосходит WAF ModSecurity по результативности обнаружения атак подбора пароля и CSRF атак.

4.3 Сценарий 3 – Моделирование эксплуатации 0-day уязвимости

Разработка атаки нулевого дня является крайне сложной задачей. Однако на практике для сигнатурных средств защиты возможно смоделировать ситуацию, когда одна из реализаций атаки становится атакой нулевого дня. Идея состоит в том, чтобы «откатить назад» («деградировать») базу решающих правил «назад по истории» и исключить одну из сигнатур, чтобы соответствующая атака стала атакой нулевого дня.

Для этого был проанализирован публичный репозиторий базы правил CRS, и обнаружен запрос от сообщества (issue #2211) на включение новой сигнатуры в базу с тегом «False Negative – Evasion». Заголовок сообщения: «Drop keyword not blocked for SQL injection» . Суть сообщения авторам проекта CRS заключалась в том, что одна из атак типа «SQL инъекция» не обнаруживалась WAF ModSecurity. Ошибка состояла в том, что для SQL инструкции DROP отсутствовал шаблон поиска в правиле 942360. Следующий запрос не блокировался до обновления набора правил: «https://some.web.site/index.html?q='drop table users;--» (далее – тестовая SQL инъекция нулевого дня).

В используемом в эксперименте наборе правил CRS версии 3.3.2 правило 942360 «SQL Injection» было изменено – искусственно была выполнена замена правила на предыдущую версию (более раннюю редакцию). Таким образом смоделирована ситуация, когда одна из атак стала реализацией 0-day уязвимости – после изменения в базе правил CRS стала отсутствовать информация об одной реализации SQL инъекции.

Как и предполагалось, после модификации базы решающих правил («откат назад» к старой редакции правила 942360) сигнатурный классификатор WAF ModSecurity перестал обнаруживать реализацию тестовой SQL инъекции нулевого дня в отношении объекта защиты DVWA (табл. 4).

При этом обученная модель типа «случайный лес» успешно обнаруживала атаку – эксплуатацию искусственной 0-day уязвимости. Важно отметить, что конкретная реализация тестовой SQL инъекции нулевого дня не предъявлялась модели на этапе обучения. И обнаружение ранее неизвестной атаки стало возможным благодаря обобщающей способности модели машинного обучения.

Табл. 4. Реакция сравниваемых средств защиты при эксплуатации 0-day уязвимости
Table 4. Response of compared security systems to the exploitation of the zero-day vulnerability

Средство защиты	**Тип атаки и результат обнаружения**	**Примечание**
	SQL Injection	
NIDS Suricata	–	Атаки в зашифрованном трафике HTTPS не обнаруживаются.
WAF ModSecurity	–	Реализация SQL атаки (правило для обнаружения которой преднамеренно исключено из базы правил) не обнаруживается.
ML COB	+	SQL-атака обнаруживается. Конкретная атака не воспроизводилась на этапе обучения, обобщающая способность модели позволяет обнаруживать ранее неизвестные атаки (0-day).

Проведенный эксперимент показал следующее:

- NIDS Suricata не обнаруживает атаки сценария 3 в зашифрованном трафике HTTPS;
- качество обнаружения WAF ModSecurity зависит от полноты базы решающих правил, а ML COB – от качества обучающего трафика;
- ML COB позволяет обнаруживать ранее неизвестные атаки (0-day).

4.4 Сценарий 4 – Комплексное сравнение эффективности средств защиты

Для оценки общей эффективности средств защиты был проведен эксперимент, в ходе которого моделировались 6 типов атак и 2 вида фонового трафика. Информация по типам трафика и средствам генерации представлена в табл. 5.

Табл. 5. Информация по типам трафика и средствам генерации
Table 5. Information about traffic types and generation tools

Тип трафика		**Средство моделирования**
Атаки	XSS	xsser
	Web Shell	weevely
	OS Command Injection	commix
	SQL Injection	sqlmap
	CSRF	browser, hackapp
	Brute Force	patator
Фоновый трафик	Обычный трафик работы пользователя в браузере	browser
	Обычные запросы пользователя, содержащие специальные символы (фразы), на которые могут реагировать сигнатурные средств защиты	OWASP ZAP

Фоновый трафик был представлен двумя подклассами:

- фоновый трафик, формируемый по результатам штатной работы пользователя в браузере;
- фоновый трафик, формируемый по результатам штатной работы пользователя в браузере с добавлением в обмен с сервером специальных символов (фраз), на которые реагировал WAF ModSecurity на 5 уровне паранойи (Paranoia Level).

Наличие второго подкласса фонового трафика позволяет рассчитать оценки показателей качества классификаторов, связанные с ложным срабатыванием сигнатурных средств защиты информации. В качестве спецсимволов (фраз) использовались следующие: \1 |2 /3 (4 }5 @6 "7 <8 >9 <>10 !11 ?12 #13 $14 %15 ^16 ~17 +18 {19 }20 <script21 <SCRIPT22 <Script23 <SCRIPt24 <sCRIPT25 и др.

Генерация такого трафика осуществлялась с помощью OWASP ZAP путем фаззинга запроса GET https://dvwa.isp/vulnerabilities/ xss_r/?name=$$$$$$$, где вместо $$$$$$$ из файла поочередно подставлялись отобранные спецсимволы (фразы).

Полученные по результатам проведенного эксперимента частные показатели качества работы средств защиты в отношении каждого типа трафика представлены в табл. 6. Итоговые результаты проведенного комплексного тестирования представлены в табл. 7.

Табл. 6. Частные показатели качества работы средств защиты
Table 6. Partial indicators of the quality of the security systems' performance

Средство защиты	**Атаки (1987 атак)**												**Фоновый трафик (2000 сессий)**			
	XSS (xsser; 1291 атак)		**Web Shell (weveely; 64 атаки)**		**Command Injection (commix; 197 атак)**		**SQL Injection (sqlmap; 285 атак)**		**CSRF (browser, hackapp; 100 атак)**		**Brute Force (patator; 50 атак)**		**Обычный трафик (browser; 1000 сессий)**		**Трафик, содержащий спецсимволы (ZAP; 1000 сессий)**	
	TP	**FN**	**TP**	**FN**	**TP**	**FN**	**TP**	**FN**	**TP**	**FN**	**TP**	**FN**	**TN**	**FP**	**TN**	**FP**
Suricata (HTTPS)	0	1291	0	64	0	197	0	285	0	100	0	50	1000	0	1000	0
Suricata (HTTP)	269	1022	0	64	10	187	125	160	0	100	0	50	1000	0	1000	0
ModSecurity (Paranoia Level = 1)	1257	34	49	15	182	15	276	9	0	100	0	50	1000	0	800	200
ModSecurity (Paranoia Level = 2)	1266	25	59	5	197	0	276	9	0	100	0	50	1000	0	680	320
ModSecurity (Paranoia Level = 3)	1266	25	61	3	197	0	276	9	0	100	0	50	1000	0	680	320
ModSecurity (Paranoia Level = 4)	1268	23	63	1	197	0	278	7	0	100	0	50	1000	0	40	960
ModSecurity (Paranoia Level = 5)	1269	22	63	1	197	0	278	7	0	100	0	50	1000	0	0	1000
ML COB	1288	3	64	0	197	0	279	6	92	8	50	0	1000	0	880	120

TP (true positive) – атака классифицируется как атака;
FN (false negative) – атака классифицируется как фоновый трафик;
FP (false positive) – фоновый трафик классифицируется как атака;
TN (true negative) – фоновый трафик классифицируется как фоновый трафик.

Табл. 7. Итоговые результаты комплексного тестирования средств защиты
Table 7. Final results of the comprehensive testing of the security systems

Средство защиты	**TP**	**FN**	**TN**	**FP**	**Precision (Точность)**	**Recall (Полнота)**	**Specificity (Специфич-ность)**	**F1 (F-мера)**	**Accuracy (Доля правильных ответов)**
Suricata (HTTPS)	0	1987	2000	0	--	0	1	--	0.502
Suricata (HTTP)	404	1583	2000	0	1	0.203	1	0.338	0.603
ModSecurity (Paranoia Level = 1)	1764	223	1800	200	0.898	0.888	0.9	0.893	0.894
ModSecurity (Paranoia Level = 2)	1798	189	1680	320	0.849	0.905	0.84	0.876	0.872
ModSecurity (Paranoia Level = 3)	1800	187	1680	320	0.849	0.906	0.84	0.877	0.873
ModSecurity (Paranoia Level = 4)	1806	181	1040	960	0.653	0.909	0.52	0.760	0.714
ModSecurity (Paranoia Level = 5)	1807	180	1000	1000	0.644	0.909	0.5	0.754	0.704
ML COB	1970	17	1780	220	0.943	0.991	0.94	0.966	0.966

Precision = TP / (TP + FP)
Recall = TP / (TP + FN)
Specificity = TN / (TN + FP)
F1 = 2 * (Precision * Recall) / (Precision + Recall)
Accuracy = TP + TN / (TP + TN + FP + FN)

Проведенный эксперимент показал, что ML COB в тестовых условиях по всем показателям превосходит качество работы NIDS Suricata и сопоставима с качеством работы WAF ModSecurity. При этом WAF ModSecurity предпочтительно использовать на первом уровне паранойи, обеспечивающем минимальное количество ложных срабатываний.

5. Ограничения использования ML COB

В представленном исследовании обнаружение веб-атак, в первую очередь, было основано на идентификации типа трафика, сгенерированного соответствующим программным средством (браузером или генератором атак типа sqlmap, xsser, patator, wevely, commix).

В ходе проведенных дополнительных экспериментов было установлено, что качество выявления атак ML COB зависит от временных характеристик передачи данных в канале. Особенно это касается атак, реализуемых с использованием браузера (например, атаки XSS, CSRF, Command Injection, SQL Injection).

В табл. 8 представлены оценки качества работы ML COB на каналах с разными временными задержками. Их моделирование осуществлялось с использованием инструмента Netem [15]. Оценка производилась на основе расчета средневзвешенных метрик по всем классам атак. При этом для атак, реализуемых через браузер, таких как CSRF и XSS, дополнительно рассчитывались частные значения метрики Recall.

Табл. 8. Оценки качества работы ML COB на каналах с разными временными задержками
Table 8. Assessment of the quality of ML IDS' performance over the channels with different time delays

Метрики / Атаки	**Обычный канал**	**Канал с внесением межпакетных задержек 300 мс**	**Канал с внесением межпакетных задержек 500 мс**	**Канал с внесением межпакетных задержек 800 мс**
Средневзвешенные метрики по всем классам атак				
Precision	0.981	0.973	0.931	0.828
Recall	0.981	0.975	0.938	0.786
Specificity	0.999	0.999	0.999	0.999
F1	0.981	0.972	0.928	0.757
Accuracy	0.981	0.975	0.938	0.786
Метрика Recall для атак, реализуемых через браузер (CSRF, XSS)				
CSRF	0.92	0	0	0
XSS	0.93	0	0	0

Как видно из табл. 8, качество работы модели, обученной только на данных, собранных на реальной сети в обычных условиях функционирования, снижается при внесении задержек в трафик. При этом атаки CSRF и XSS не выявляются вообще. Это связано с тем, что трафик, формируемый в результате реализации атак, осуществляемых через браузер, очень похож на фоновый, т.е. на обычную работу пользователя в браузере. В связи с этим изменения временных характеристик передачи данных минимизируют изначальные отличия зловредных действий от обычных. На рис. 2 представлен пример зависимости основных параметров сессий браузера от задержек в действиях пользователя.

Из рисунка видно, что внесение пауз приводит к тому, что сессии, содержащие зловредную нагрузку, изменяют свои основные параметры (продолжительность и размер передаваемых данных) и становятся не свойственными для класса атак, сформированного по результатам обучения. В этой связи необходимо отметить, что выявление такого типа атак является сложной задачей, решение которой может быть получено только в отношении конкретного веб-приложения при точном знании ограничений по входным и выходным данным страниц и максимально полном моделировании передачи зловредной нагрузки.

Опираясь на данные табл. 8 и рис. 2, можно сделать вывод о том, что включение в обучающий набор данных трафика с задержками позволит повысить качество и устойчивость ML COB в условиях возможного изменения временных характеристик передачи данных (реализации состязательных атак).

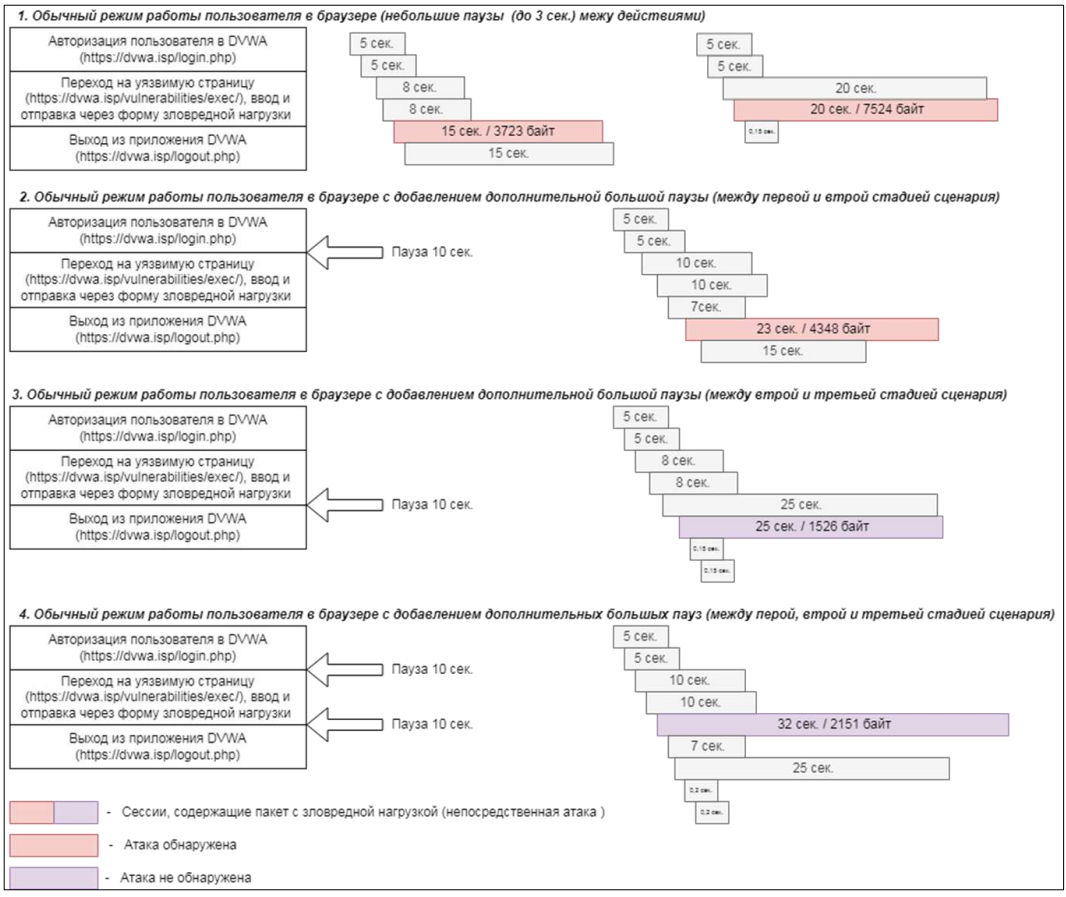


Рис. 2. Пример зависимости основных параметров сессий от задержек в канале
Fig. 2. Example of the dependency of the sessions' main parameters on the delays in the channel

В рамках текущего исследования был проведен эксперимент, в ходе которого исходная модель машинного обучения была дообучена на данных трафика с задержками 500 мс. Полученные оценки качества работы дообученной ML СОВ представлены в табл. 9.

Табл. 9. Оценки качества работы дообученной ML СОВ на каналах с разными временными задержками
Table 9. Assessment of the quality of ML IDS' performance over the channels with different time delays after additional training

Метрики / Атаки	**Обычный канал**	**Канал с внесением межпакетных задержек 300 мс**	**Канал с внесением межпакетных задержек 500 мс**	**Канал с внесением межпакетных задержек 800 мс**
Средневзвешенные метрики по всем классам атак				
Precision	0.982	0.993	0.995	0.981
Recall	0.982	0.993	0.995	0.980
Specificity	0.999	0.999	0.999	0.999
F1	0.982	0.993	0.995	0.980
Accuracy	0.982	0.993	0.995	0.980
Метрика Recall для атак, реализуемых через браузер (CSRF, XSS)				
CSRF	0.92	0.77	0.95	0.88
XSS	0.91	0.63	0.94	0.64

Как видно из табл. 9, после дообучения модели качество ее работы повышается, в том числе и на трафике с другими задержками. При этом атаки CSRF и XSS могут быть выявлены с приемлемыми показателями качества.

Таким образом, ML СОВ эффективны, в первую очередь, в отношении идентификации программных средств генерации трафика соответствующего типа. При этом подтверждается целесообразность дообучения моделей на трафике всех доступных и появляющихся новых генераторов атак. Указанное направление представляется одним из наиболее актуальных направлений развития ML СОВ.

Включение в обучающий набор данных трафика генераторов атак позволит качественно выявлять тип используемого генератора, тип реализуемой с его помощью атаки, а также атаки, осуществляемой с помощью ранее неизвестного программного средства.

Кроме того, является целесообразным при формировании обучающего набора данных дополнительно включать в него трафик с искусственно внесенными задержками с разными интервальными значениями, что позволит повысить качество работы модели и ее устойчивость к атакам уклонения, реализуемых за счет манипуляции межпакетными задержками.

В табл. 10 представлены основные достоинства и недостатки ML СОВ и рассматриваемых сигнатурных средств защиты.

Табл. 10. Достоинства и недостатки рассматриваемых средств защиты
Table 10. Advantages and disadvantages of the discussed security systems

Средство защиты	Достоинства	Недостатки
NIDS Suricata	1. Обнаружение атак в режиме, близком к реальному. 2. Простота интерпретации данных о выявлении атак на основе анализа сработавшей сигнатуры. 3. Независимость от параметров сетевого соединения.	1. Работа только на открытом трафике. 2. Зависимость от полноты и качества базы решающих правил. 3. Невозможность выявления ранее неизвестных атак.
WAF ModSecurity	1. Обнаружение атак в режиме, близком к реальному. 2. Простота интерпретации данных о выявлении атак на основе анализа сработавшей сигнатуры. 3. Независимость от параметров сетевого соединения.	1. Зависимость от полноты и качества базы решающих правил. 2. Невозможность выявления ранее неизвестных атак.
ML СОВ	1. Обнаружение атак в режиме, близком к реальному. 2. Возможность выявления ранее неизвестных атак. 3. Возможность работы на зашифрованном трафике.	1. Сложность анализа результатов срабатывания ML СОВ (атакой признается вся сессия без информации о переданной полезной нагрузке). 2. Зависимость от полноты и качества обучающего набора данных. 3. Зависимость от характеристик канала связи, трафик которого использовался при формировании обучающего набора данных. 4. Подверженность атакам на ML системы. 5. Необходимость иметь возможность атаковать реальный объект защиты на этапе обучения.

Учитывая полученные оценки качества работы ML СОВ, а также выявленные их достоинства и недостатки, можно сделать вывод о том, что системы такого класса могут использоваться как самостоятельно, так и совместно с сигнатурными средствами. При этом целесообразным является их использование в качестве дополнения к существующим сигнатурным средствам защиты, что в комплексе позволит повысить общую эффективность выявления атак, в том числе и ранее неизвестных атак.

6. Заключение

Представленный в исследовании подход к сравнению систем обнаружения вторжений на основе нескольких независимых сценариев и комплексного тестирования позволил выявить основные достоинства и недостатки ML COB и определить ограничения в практическом применении таких систем.

Представлены реальные практические условия, при которых система обнаружения вторжений на основе машинного обучения превосходит сигнатурные средства защиты по качеству обнаружения атак.

При работе с зашифрованным трафиком HTTPS продемонстрировано превосходство ML COB по сравнению с сигнатурной COB Suricata. Следует отметить, что это преимущество теряет значение в условиях возможного принудительного расшифровывания трафика HTTPS (SSL Bumping).

На узком классе веб-атак ML COB показала сопоставимое качество обнаружения атак с сигнатурным межсетевым экраном веб-приложений ModSecurity. При этом в выбранных условиях эксперимента ML COB представляла собой самостоятельное решение, не требующее наличия сигнатурного анализатора.

Основным достоинством ML COB является возможность выявления ранее неизвестных атак и работа на зашифрованном трафике. Среди недостатков систем такого класса необходимо отметить зависимость качества обнаружения атак ML COB от полноты и качества обучающего набора данных, который должен учитывать возможности изменения параметров сетевых соединений и реализации других состязательных атак со стороны потенциального нарушителя.

В общем случае для реализации защиты от широкого перечня атак с максимальным качеством представляется наиболее рациональным применение гибридных систем обнаружения атак, сочетающих в себе и возможности сигнатурного анализа, и эвристические методы обнаружения, в том числе, основанные на машинном обучении.

Список литературы / References

[1] Актуальные киберугрозы: итоги 2021 года / Actual cyber threats: results of 2021. Available at: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/Cybersecurity_threatscape_2021_RUS.pdf, accessed 03.11.2022 (in Russian).

[2] Appelt D., Nguyen C.D., Panichella A., Briand L.C. A Machine-Learning-Driven Evolutionary Approach for Testing Web Application Firewalls. IEEE Transactions on Reliability, vol. 67, no. 3, Sept. 2018, pp. 733-757.

[3] Betarte G., Giménez E., Martínez R., Pardo Á. Machine learning-assisted virtual patching of web applications. 2018, 14 p. DOI: 10.48550/arXiv.1803.05529.

[4] Applebaum S., Gaber T., Ahmed A. Signature-based and Machine-Learning-based Web Application Firewalls: A Short Survey. Procedia Computer Science, vol. 189, 2021, pp. 359-367..

[5] Mohammadi S., Namadchian A. Anomaly-based Web Attack Detection: The Application of Deep Neural Network Seq2Seq with Attention Mechanism. The ISC International Journal of Information Security, vol. 12, issue 1, 2020, pp. 44-54.

[6] Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Синтез модели машинного обучения для обнаружения компьютерных атак на основе набора данных CICIDS2017. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 81-94 / Goryunov M.N., Matskevich A.G., Rybolovlev D.A. Synthesis of a Machine Learning Model for Detecting Computer Attacks Based on the CICIDS2017 Dataset. Trudy ISP RAN/Proc. ISP RAS, 2020, vol. 32, issue 5, pp. 81-94 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–6.

[7] Damn Vulnerable Web Application (DVWA). Available at: https://github.com/digininja/DVWA, accessed 3.11.2022.

[8] ModSecurity. Available at: https://github.com/SpiderLabs/ModSecurity, accessed 3.11.2022.

[9] OWASP ModSecurity Core Rule Set (CRS). Available at: https://github.com/coreruleset/coreruleset, accessed 3.11.2022.

[10] Suricata. Available at: https://github.com/OISF/suricata, accessed 3.11.2022.
[11] Proofpoint Emerging Threats Rules. Available at: https://rules.emergingthreats.net/open/, accessed 3.11.2022.
[12] HTTPS encryption on the web – Google Transparency Report 2021. Available at: https://transparencyreport.google.com/https/overview, accessed 3.11.2022.
[13] Weevely. Available at: https://github.com/PhHitachi/Weevely, accessed 3.11.2022.
[14] Patator. Available at: https://github.com/lanjelot/patator, accessed 3.11.2022.
[15] Netem. Available at: https://github.com/hansfilipelo/netem, accessed 3.11.2022.

Информация об авторах / Information about authors

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент ВШЭ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.

Aleksandr Igorevich GETMAN – Ph.D. in physical and mathematical sciences, senior researcher at ISP RAS, associate professor at HSE. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.

Максим Николаевич ГОРЮНОВ – кандидат технических наук, сотрудник Академии ФСО России. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы анализа защищенности, машинное обучение, безопасная разработка программного обеспечения.

Maxim Nikolaevich GORYUNOV – Ph.D., employee, the Academy of Federal Security Guard Service of the Russian Federation. Research interests: information security, intrusion detection systems, security analysis systems, machine learning.

Андрей Георгиевич МАЦКЕВИЧ – кандидат технических наук, доцент, сотрудник Академии ФСО России. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы антивирусной защиты, машинное обучение, криптографические методы защиты информации.

Andrey Georgievich MATSKEVICH – Ph.D., docent, employee, the Academy of Federal Security Guard Service of the Russian Federation. Research interests: information security, intrusion detection systems, anti-virus protection systems, machine learning, cryptographic methods for protecting information.

Дмитрий Александрович РЫБОЛОВЛЕВ – кандидат технических наук, сотрудник Академии ФСО России. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, машинное обучение, криптографические методы защиты информации.

Dmitry Aleksandrovich RYBOLOVLEV – Ph.D., employee, the Academy of Federal Security Guard Service of the Russian Federation. Research interests: information security, intrusion detection systems, machine learning, cryptographic methods for protecting information.

DOI: 10.15514/ISPRAS-2022-34(5)-8

Определение влиятельных пользователей социальной сети по двудольному графу комментариев

[1] *Р.К. Пастухов, ORCID: 0000-0001-6919-0701 <pastukhov@ispras.ru>*
[1]*М.Д. Дробышевский, ORCID: 0000-0002-1639-9154 <drobyshevsky@ispras.ru>*
[1,2] *Д.Ю. Турдаков, ORCID: 0000-0001-8745-0984 <turdakov@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*

Аннотация. С развитием онлайновых социальных сетей все большую актуальность приобретают задачи выявления пользователей, которые оказывает большое влияние на других участников социальных сетей. Важным источником информации служат данные о комментировании пользователями контента, создаваемого другими пользователями. В работе предлагается метод определения влиятельности, основанный на двудольном графе пользователь-комментарий-контент, а также использующий информацию о текстовых сообщениях и реакции на них других пользователей. Кроме того, предложен метод выявления сообществ пользователей в таком графе на основе общих интересов. Результаты тестирования на коллекциях данных сетей ВКонтакте и YouTube показывают корреляцию активности и влиятельности пользователя, однако самые активные комментаторы не обязательно являются самыми влиятельными. Анализ сообществ показывает положительную корреляцию между размером сообщества, числом наиболее влиятельных пользователей в нем и средней влиятельностью пользователей сообщества.

Ключевые слова: социальные сети; влиятельные пользователи; двудольный граф; поиск сообществ.

Для цитирования: Пастухов Р.К., Дробышевский М.Д., Турдаков Д.Ю. Определение влиятельных пользователей социальной сети по двудольному графу комментариев. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 127-142. DOI: 10.15514/ISPRAS-2022-34(5)-8

Благодарности: Исследование выполнено в рамках научной программы Национального центра физики и математики (проект №9 “Искусственный интеллект и большие данные в технических, промышленных, природных и социальных системах”).

Detecting Influential Users in Social Networks Based on Bipartite Comments Graph

[1] *R.K. Pastukhov, ORCID: 0000-0001-6919-0701 <pastukhov@ispras.ru>*
[1] *M.D. Drobyshevskiy, ORCID: 0000-0002-1639-9154 <drobyshevsky@ispras.ru>*
[1,2] *D.Yu. Turdakov, ORCID: 0000-0001-8745-0984 <turdakov@ispras.ru>*

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

Abstract. With the development of online social networks, the task of identifying users who have a great influence on other participants in social networks is becoming increasingly important. An important source of information is user comments on content created by other users. The paper proposes a method for determining influence based on a bipartite user-comment-content graph. It incorporates information about text messages and the reactions of other users to them. In addition, we propose a method for identifying user communities in such a graph based on common interests. Experiments on data collections from VKontakte and YouTube networks show the correlation between user activity and influence, however, the most active commenters are not necessarily the most influential. Community analysis shows a positive correlation between the size of a community, the number of most influential users in it, and the average influence of community users.

Keywords: social network; influential user; bipartite graph; community detection

For citation: Pastukhov R.K., Drobyshevskiy M.D., Turdakov D.Yu. Detecting Influential Users in Social Networks Based on Bipartite Comments Graph. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 127-142 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-8

Acknowledgements. This work was done under a scientific program of National center of physics and mathematics (project no. 9 “Artificial intelligence and big data in technical, industrial, natural, and social systems”.

1. Введение

Социальные сети становятся все более обширным источником информации о социуме, поведении людей, их предпочтениях, общественных трендах. Пользователи оставляют информацию о себе, устанавливают связи с другими пользователями, пишут сообщения, комментируют контент, созданный другими пользователями, ставят оценки и т.д. Для извлечения полезной информации из всех этих массивов данных последние годы разрабатываются множество подходов и алгоритмов их анализа.

В данной работе ставятся следующие задачи:

1) разработать метод выявления влиятельных пользователей сетевых сообществ на основании анализа активности пользователей соцсетей, включая анализ их сообщений;
2) разработать метод выделения сетевых сообществ пользователей на основании анализа их активности;
3) провести экспериментальные исследования разработанных методов на коллекциях данных, собранных из ВКонтакте и YouTube.

Разрабатываемые методы выделения сетевых сообществ на основании анализа активности пользователей соцсетей должны на основе анализа разных типов взаимодействия пользователей (комментарии, лайки, репосты и т.п.) выделять неявные сообщества (группы) наиболее активных пользователей соцсетей, оказывающих влияние на других пользователей соцсетей. Для предварительных экспериментов взяты две коллекции данных, собранных на основе нескольких групп социальной сети ВКонтакте и нескольких каналов YouTube.

В работе проведен обзор методов поиска сообществ и определения влиятельности пользователей социальных сетей. Разработан метод подсчета влиятельности пользователей в

специфике имеющихся данных, а также предложен метод поиска сообществ. Выполнено тестирование разработанных методов на двух коллекциях данных.

2. Обзор литературы

В задаче необходимо с одной стороны определить сообщества пользователей социальной сети, с другой стороны, выделить наиболее активных и влиятельных ее членов. Приведем далее обзор методов определения структуры сообществ в социальных сетях, а затем рассмотрим подходы и алгоритмы поиска влиятельных вершин.

2.1 Методы определения сообществ пользователей в социальных сетях

Пользователи онлайн социальных сетей не всегда явно указывают сообщества, к которым они принадлежат. Кроме того, некоторые сообщества могут возникать неявно, например, коллеги, одноклассники, жильцы дома и т.п. Данные социального графа являются богатым источником информации для поиска явных и неявных сообществ, их анализа и дальнейшего использования. С ростом популярности онлайн социальных сетей последние годы активно появляются и развиваются методы поиска скрытой структуры сообществ. Большинство методов основываются только на социальных связях, которые отражены в виде связи дружбы, подписки и т.д. и в социальном графе выражаются ребрами, соединяющими соответствующих пользователей.

Можно выделить три основных класса алгоритмов, показывающих наиболее высокое качество определения сообществ. Это подходы на основе локальной оптимизации, вероятностные методы и методы распространения меток. Рассмотрим подробнее представителей этих классов, а также некоторые другие подходы.

2.1.1 Методы на основе локальной оптимизации

Подход заключается в оптимизации некоторой локальной функции качества для каждого сообщества независимо. Например, задается ядро будущего сообщества и постепенно расширяется путем добавления новых смежных вершин. В методе GCE [1] задается следующая функция качества:

$$F_S = \frac{k_{\text{in}}^S}{(k_{\text{in}}^S + k_{out}^S)^\alpha}.$$

При этом S – индуцированный подграф, являющееся растущим ядром сообщества. k_{in}^S – удвоенное число ребер, которые имеют начало и конец в S, k_{out}^S – число ребер с одной вершиной в S. α – параметр метода.

Другой популярный метод – OSLOM [2]. Метод способен обнаруживать сообщества в сетях с учетом направлений ребер, весов ребер, перекрывающихся сообществ, иерархии и динамики сообщества. Он основан на локальной оптимизации функции пригодности, выражающей статистическую значимость кластеров по отношению к случайным флуктуациям, которая оценивается с помощью инструментов экстремальной и порядковой статистики. OSLOM можно использовать отдельно или в качестве процедуры уточнения сообществ, обнаруженных другими методами.

2.1.2 Вероятностные модели

В подходах этого типа задается параметрическая модель, отражающая распределение на множестве графов с фиксированным числом вершин. Обычно в модель включается матрица связи вершин и сообществ. Работа алгоритма заключается в максимизации правдоподобия исходного графа в рамках модели. В итоге получается оптимальная матрица инцидентности вершин и сообществ. Примерами таких алгоритмов являются MOSES [3], BCD [4], BIGCLAM [5], CESNA [6].

Отдельно можно выделить методы максимизации модулярности. Изначально модулярность была определена Ньюманом и Гирваном [7] для измерения качества разбиения графа на набор кластеров как доля ребер внутри кластеров за вычетом ожидаемой такой доли в случайном графе с теми же вершинами и их степенями:

$$Q = \frac{1}{2E}\sum_{c\in C}\sum_{i,j\in c}\left(A_{ij} - \frac{k_i k_j}{2E}\right).$$

Здесь A_{ij} обозначает матрицу смежности графа, C – набор сообществ (покрытий), k_i – степень вершины i. Численное значение характеризует степень конкретной кластеризации графа. Она высока, когда кластеры плотные и слабо связаны друг с другом, и низка, когда кластеры формируются случайным образом.

В литературе было предложено множество эвристических методов для нахождения разбиений вершин графа с высокой модулярностью за разумное время. К таким подходам относятся жадные алгоритмы, спектральные методы, экстремальная оптимизация, имитация отжига, метод выборки и математическое программирование – подробнее можно найти обзор в работе [8]. В той же работе описан настраиваемый алгоритм обнаружения сообществ, который итеративно улучшает показатель качества сообщества Q путем разделения и слияния заданной структуры сетевого сообщества. Другой метод [9] представляет собой усовершенствованную версию популярного метода Louvain [10] для работы с направленными графами на основе расширения понятия модулярности для направленных графов.

2.1.3 Методы распространения меток

Идея распространения меток по ребрам графа популярна для графов с очень большим числом вершин и ребер благодаря низкой вычислительной сложности, хорошей масштабируемости и простоты реализации алгоритмов. Основная схема методов данного класса – итеративный процесс рассылки и получения вершинами графа меток сообществ вдоль ребер. На каждом шаге вершина накапливает входящие метки и отправляет свою метку соседям. Обычно отправка меток происходит синхронно и независимо, но существуют также и асинхронные схемы. Типичными представителями класса являются методы SLPA [11], BMLPA [12] и COPRA [13], которые отличаются только способом инициализации меток и стратегиями их обмена.

2.1.4 Методы, основанные на эго-сообществах

Существует ряд методов, основанных на понятии эго-сети пользователя – объединении вершин и ребер первой окрестности некоторой вершины. При этом рассматривается набор сообществ, которым принадлежат вершины эго-сети. Алгоритмы предлагают различные способы агрегации эго-сообществ в более большие. В методе Node Perception [14] строится мета-сеть эго-сообществ, в которой затем производится кластеризация. Алгоритмы DEMON [15] и EgoClustering [16] являются его модификациями. Метод EgoLP [17] использует эго-сообщества для агрегации и фильтрации меток сообществ, распространяющихся между узлами. Благодаря хорошей масштабируемости распределенная реализация EgoLP способна обрабатывать графы с миллиардами вершин.

2.2 Методы поиска влиятельных пользователей в социальных сетях

Постоянно растущее количество информации, обращающееся в социальных сетях, заставляет участников этих сетей конкурировать за внимание и влияние, полагаясь на других людей для распространения своих сообщений. Популярность и статус участников социальных сетей измеряются уровнем внимания, которое они получают в виде подписчиков, которые создают ссылки на свои учетные записи для автоматического получения контента, который они создают. Другим аспектом является влияние, которым обладают эти люди, которое

определяется фактическим распространением их контента через сеть. Это влияние определяется многими факторами, такими как новизна и резонанс их сообщений с сообщениями их подписчиков, а также качество и частота контента, который они генерируют.

Для выявления наиболее влиятельных пользователей сети в литературе предложены разные подходы в зависимости от размеров сети, имеющейся информации о структуре сети, дополнительном знании и содержании сообщений пользователей и других факторов. Далее рассмотрим следующие категории подходов: методы поиска хабов, методы использующие известную структуру графов, методы, учитывающие временную динамику, и наконец методы, использующие текстовые сообщения пользователей.

2.2.1 Методы поиска хабов

Известно, что по сравнению с обычными пользователями влиятельные пользователи [18] играют более важную роль в социальных графах с точки зрения связности, влияния и распространения информации. Они задают тенденцию в оценке или отношении к тому или иному факту или событию. Вследствие этого анализ структуры связности и обмена информацией только во “влиятельной” части сети, который содержит всех влиятельных пользователей и их попарные отношения, дает ряд перспективных возможностей.

Влиятельность вершины можно определить, как меру ее центральности в графе. Чаще всего используется степень, то есть количество связей с другими вершинами. Предполагается, что люди с большим количеством подписчиков или друзей, так называемые хабы, в социальной сети имеют широкий охват аудитории, а значит, могут формировать мнение других пользователей.

Большинство современных социальных сетей – Фейсбук, Твиттер, Инстаграм, Вконтакте – не позволяют напрямую скачать полную структуру социального графа для проведения анализа. Поэтому разработан ряд методов поиска хабов в больших сетях на основе знания окрестности текущего узла. В алгоритме взвешенного случайного блуждания [19] необходимо знать степени соседей посещаемого узла на каждом шаге алгоритма, чтобы обнаружить узлы с наивысшей степенью. В нескольких работах [20, 21] исследуется алгоритм максимальной наблюдаемой степени (MOD – Maximal observed degree). Стратегия обхода выбирает узел с наивысшей наблюдаемой степенью на каждом шаге. Метод DE-Crawler [22] представляют комбинирует две основные стратегии: MOD и случайные блуждания. Первая отвечает за уплотнение, когда алгоритм стремится найти как можно больше узлов в текущей плотной области (сообществе). Фаза блужданий применяется для расширения, когда необходимо переместиться в другую плотную область. Авторы показывают, что предложенный метод лучше всего подходит для решения задачи обнаружения как можно большего количества узлов.

Показано, что метод случайного блуждания также эффективен для быстрого нахождения узлов большой степени в очень больших графах [23]. Экспериментально показано что метод с высокой вероятностью находит топ списка узлов с хорошей точностью. В другой работе предлагается рандомизированный двухэтапный алгоритм [23] определения топ-k узлов с высокой точностью, используя малый (например, 1000) бюджет запросов. Например, для социальной сети Twitter, имеющей более миллиарда пользователей, они определяют топ-100 пользователей с наибольшим количеством подписчиков с точностью более 90%. Позднее предложена более усовершенствованная версия – трехшаговый алгоритм [24].

2.2.2 Поиск влиятельных вершин в графах с полной информацией

При наличии полной информации о социальном графе значительно расширяется возможный инструментарий для анализа поведения пользователей. Исследования распространения информации в Twitter показали [25], что большинство пользователей действуют как пассивные потребители информации и не передают контент в сеть. Следовательно, чтобы

люди стали влиятельными, они должны не только привлечь внимание и, но и преодолеть пассивность пользователей. В работе [25] предлагается алгоритм, определяющий влияние и пассивность пользователей на основе их активности по передаче информации. Оценка, проведенная с набором данных о 2,5 миллионах пользователей, показывает, что вычисленная мера влиятельности является хорошим предсказателем кликов по URL-адресам, превосходит некоторые другие меры, которые явно не учитывают пассивность пользователей. Кроме того, авторы показывают, что высокая популярность не обязательно означает высокое влияние, и наоборот.

Предложенная модель делает следующие предположения:

1) оценка влияния пользователя зависит от количества людей, на которых он влияет, а также от их пассивности;
2) оценка влияния пользователя зависит от того, насколько вовлечены люди, на которых он влияет. Вовлеченность измеряется количеством внимания, которое пользователь уделяет данному пользователю по сравнению со всеми остальными;
3) оценка пассивности пользователя зависит от влиятельности тех, с кем он контактирует, но не находится под влиянием;
4) оценка пассивности пользователя зависит от того, насколько он отвергает влияние другого пользователя по сравнению со всеми остальными.

Алгоритм влияния-пассивности итеративно одновременно вычисляет показатели пассивности и влияния следующим образом. Задан граф G с вершинами N, ребрами E и их весами W. Веса w_{ij} на ребре $e = (i, j)$ представляет собой отношение влияния i на j к общему влиянию, которое i пытается оказать на j. Алгоритм выводит функцию $I: N \to [0,1]$, которая представляет относительное влияние узла на сеть, и функцию $P: N \to [0,1]$, которая представляет относительную пассивность узла.

Для каждого ребра (i, j) определяется скорость принятия как

$$u_{ij} = \frac{w_{ij}}{\sum_{k:(k,j)\in E} w_{kj}}.$$

Это значение представляет собой количество влияния, которое пользователь j принял от пользователя i, нормализованное по общему влиянию, принятому j от всех пользователей в сети. Уровень принятия можно рассматривать как вовлеченность или лояльность пользователя j пользователю i. С другой стороны, для каждого $e = (j, i) \in E$ определяется скорость отказов, как

$$v_{ij} = \frac{1 - w_{ji}}{\sum_{k:(j,k)\in E} \left(1 - w_{jk}\right)}.$$

Поскольку значение $\left(1 - w_{ji}\right)$ представляет собой количество влияния, которое пользователь i отвергает от j, то значение v_{ji} представляет влияние, которое пользователь i отвергает от пользователя j, нормализованное по общему влиянию, отклоненному от j всеми пользователями в сети. Алгоритм основан на следующих операциях:

$$I_i \leftarrow \sum_{j:(i,j)\in E} u_{ij} P_j$$

и

$$P_i \leftarrow \sum_{j:(j,i)\in E} v_{ji} I_j.$$

В итоге исследование показывает, что корреляция между популярностью и влиянием слабее, чем можно было бы ожидать. Это отражение того факта, что для распространения информации в сети люди должны пересылать ее другим членам, поэтому им приходится активно участвовать, а не просто пассивно читать ее.

Авторы другой работы [26] предложили модель двудольного графа с направленным взвешенным пользовательским контентом. Модель измеряет влияние, комбинируя как структурные свойства сети, так и контент, опубликованный пользователями. Итеративный алгоритм предназначен для вычисления двух показателей: влияния пользователей и охвата досок (board) в сети Pinterest. Эксперименты показывают, что модель может эффективно обнаруживать наиболее влиятельных пользователей и популярный контент.

Предложенный алгоритм основывается на идеях, подобных предыдущему алгоритму, и тоже является итерационным.

С помощью алгоритма авторы успешно определили десять самых влиятельных пользователей в наборе данных. Кроме того, хотя, как правило, влиятельные пользователи имели очень большое количество подписчиков (по сравнению со средним числом подписчиков для пользователя), не было никакой конкретной корреляции между ранжированием влиятельности и количества подписчиков. Этот результат согласуются с выводом из предыдущей работы и многими другими.

2.2.3 Поиск влиятельных сообществ пользователей

Другой подход используется в работе [27]. Рассматривается задача поиска сообществ пользователей как обнаружение скрытой структуры сообщества в больших социальных сетях на основе их общих интересов. Исходная мотивация состоит в наблюдении, что пользователи обычно уделяют больше внимания тем пользователям, у которых схожие интересы, что позволяет разделить пользователей на разные сообщества в соответствии с их общими интересами. Авторы предлагают два алгоритма (эвристический и HWeBA) для эффективного обнаружения влиятельных сообществ, которые они называют ядрами, использующих общие интересы в крупных социальных сетях.

Предварительный статистический анализ показал, что сумма весов ребер влиятельных пользователей намного больше, чем у других пользователей в социальной сети системы Epinions. Другими словами, каждый член ядра сообщества имеет более высокую сумму весов ребер (например, степень общих интересов) в/из ядра, чем вершина вне ядра.

В статье [28] рассматривается трактовка сообществ с разных точек зрения в зависимости от свойства изучаемой сети. Кроме классических структурных сообществ на основе связи дружбы и подписок выделяются сообщества, основанные на деятельности, на основе темы контента и на основе взаимодействия. Авторы анализируют набор данных Twitter, используя три различных веса структурной сети, предназначенных для выделения этих трех типов сообществ.

Поиск сообществ, основанных на деятельности, мотивируется вопросом “У каких пользователей в сети есть схожие профили активности?”. Сообщество рассматривается как пользователи, которые используют (или не используют) сервис в одно и то же время. Авторы говорят о группах пользователей, которые обладают так называемой “способностью прогнозирования на основе активности” друг друга, т.е., учитывая пользователя u и подписчика f, которые являются членами сообщества, обнаруженного с этим взвешиванием, существует снижение неопределенности в отношении профиля активности подписчика f (твиттинг или молчание), учитывая историю твитов пользователя u, игнорируя информацию, представленную в истории твитов подписчика f. Алгоритм рассматривает каждого пользователя Твиттера как единицу обработки информации, но полностью игнорирует содержание их твитов. Затем взвешиваются направленные ребра (сообщаемые отношения между подписчиками и подписчиками) между пользователями с помощью так называемой энтропии передачи, рассчитанной между историей твитов подписчика f и пользователя u.

Кратко типы сообществ можно охарактеризовать следующими вопросами:

- на основе структуры: кто ваши заявленные друзья; за кем вы следуете?
- на основе активности: кто имеет схожие профили активности?

- тематические: о чем вы говорите?
- на основе взаимодействия: с кем вы общаетесь?

Авторы показывают, что использование разных определений сообщества раскрывает разные и интересные отношения между пользователями. Кроме того, утверждается, что различные взгляды на сеть не выявляются при использовании только структурной информации о сети или какой-либо одной схемы взвешивания.

2.2.4 Использование текстовой информации

Анализ текстовой информации, создаваемой пользователями социальных сетей, чаще всего используется в задачах классификации. Например, в работе [22] рассматривалась задача предсказания пола пользователей Youtube по текстам их комментариев. Социальная сеть моделировалась двудольным графом, одна часть вершин обозначала пользователей, другая видеоролики, связи отражали факт просмотра пользователем видеоролика. Задача предсказания пола пользователей решалась алгоритмом распространения меток (известных значений пола) вместе с использованием признаков, полученных из текстов комментариев пользователей.

Из текстов извлекались следующие три группы признаков:

1) посимвольные: средняя длина комментария, соотношение заглавных букв к общему количеству букв, отношение пунктуации к общему количеству символов;
2) на основе токенов: средняя длина комментария в словах, отношение уникальных слов к общему количеству токенов, строчные униграммы с общим количеством по всем комментариям (использовалось 10 000 наиболее часто встречающихся униграмм, частоты вычислялись по отдельному набору комментариев), использование местоимений, определителей, служебных слов;
3) на основе предложений: средняя длина комментария в предложениях, средняя длина предложения в слова.

Итоговое качество предсказания пола пользователей в экспериментах было около 90%.

3. Исходные данные

При разработке решения задачи были рассмотрены две коллекции данных из социальной сети ВКонтакте и из YouTube. На основе структуры имеющихся данных был разработан метод определения влиятельности пользователей и адаптирован алгоритм поиска сообществ с учетом заданной специфики. Далее опишем подробнее структуру данных, описание алгоритма и результаты экспериментов.

3.1 Описание коллекций данных

В обоих датасетах представлен набор элементов контента со своими характеристиками и для каждого из них лента комментариев пользователей платформы. В первом случае это посты в группах ВКонтакте, во втором – страницы видеозаписей на YouTube. Структура данных естественным образом моделируется с помощью двудольного направленного графа, где вершинами являются элементы контента и пользователи, а ребрами факт комментирования пользователем элемента контента. Граф является двудольным, поскольку связей между пользователями в данных нет, элементы контента также не связаны друг с другом. Далее приведем детали по каждому датасету.

3.1.1 Датасет ВКонтакте

Коллекция представляет собой информацию о 43 751 посте из 17 групп, родственных по географическому фактору. Число уникальных пользователей, оставлявших комментарии, 64 441. Число ребер 485 262. То есть каждый пост в среднем комментирует около 11 пользователей, один пользователь в среднем пишет комментарии в 7.5 постах.

Для каждого поста имеется следующая информация: дата публикации, тип поста, текстовое содержание, информация о прикрепленных фото и/или видео, число лайков, число репостов, число просмотров и некоторые другие сведения. Также есть лента комментариев пользователей под этим постом. По комментарию известны дата, идентификатор автора, текстовое содержание, количество ответов на данный комментарий, информация о прикрепленных фото и/или видео.

3.1.2 Датасет YouTube

В коллекции собраны комментарии к 2628 видео из 12 каналов на политические темы. Число уникальных пользователей, оставлявших комментарии, 1 246 060. Число ребер 2 536 874, то есть пересечение по пользователям довольно низкое – в среднем пользователь участвует в 2 видео. В среднем под видео 965 комментариев.

Для каждого видео имеется следующая информация: заголовок, описание, дата публикации, текстовое содержание, список тегов, число просмотров, число лайков, число дислайков, число комментариев, длительность видео. Для самих комментариев доступны: дата публикации, дата обновления, текстовое содержание, имя и идентификатор автора, число лайков и ответы на комментарий.

4. Предложенные алгоритмы

4.1 Алгоритм определения влиятельности пользователей

Учитывая двудольную природу модели данных, был предложен итеративный алгоритм вычисления влиятельности пользователя, подобный описанным в обзорах [25] и [26]. Однако в рассматриваемой задаче присутствуют некоторые отличия, из-за которых напрямую применить описанные методы нельзя. Например, все ребра направлены только от пользователей к постам/видео, но не наоборот. В связи с этим была предложена следующая методика.

Для удобства далее будем называть постом любой элемент контента, как пост так и видео. Основные принципы формирования влиятельности были предложены такие.

- пользователь влиятельный, если 1) на его комментарии много отвечают и 2) он комментирует важные посты.
- пост важный, если его 1) много комментируют и 2) комментируют влиятельные пользователи.

Далее следует формализация этих принципов. Пусть $G = (U, P, E)$ – двудольный граф, где U – пользователи, P – посты, E – ребра. Важность поста p обозначим $Reach(p)$, влиятельность пользователя u как $Inf(u)$. Схема взаимного влияния постов и пользователей дается следующими рекуррентными формулами (на шаге k+1):

$$Inf^{k+1}(u) = \sum_{p \in P} Reach^k(p) \cdot rep(u,p),$$

$$Reach^{k+1}(p) = \sum_{u \in U} \in f^k(u) \cdot eng(u,p).$$

Здесь $rep(u,p)$ обозначает количество ответов на комментарий пользователя u под постом p, а $eng(u,p)$ – вовлеченность пользователя u в обсуждение под постом p, измеряемую как общую длину всех его текстовых сообщений, написанных под этим постом.

В начале первого шага алгоритма влиятельность всех пользователей равна $Inf^0(u) = 1$, важность поста равна некоторой мере его популярности $Reach^0(p) = popularity(p)$. В качестве популярности можно взять число лайков, просмотров или число репостов. Далее алгоритм на каждом шаге обновляет значения влиятельности для всех пользователей и

важности для всех постов в соответствии с приведенными выше уравнениям. В конце каждого шага происходит нормировка значений:

$$Inf(u_i) = Inf(u_i)\frac{|U|}{\sum_k Inf(u_k)},$$

$$Reach(p_i) = Reach(p_i)\frac{|P|}{\sum_k Reach(p_k)}.$$

Для сходимости на рассмотренных датасетах алгоритму достаточно около 30 итераций, после чего ранжированный список наиболее популярных пользователей/постов перестает изменяться.

4.2 Алгоритм выявления сообществ пользователей

Поскольку явных социальных связей между пользователями в исходных данных нет, было решено использовать данные их активности в комментариях. Краеугольная идея подхода состоит в том, что если два пользователя комментируют один и тот же пост, то их интересы близки. Чем больше общих постов они комментируют, тем сильнее между ними связь. Вычислив подобные связи между любыми двумя пользователями датасета, можно выявить разбиение пользователей на группы по интересам. Для этого предлагается построить граф связей между пользователями, и провести в нем поиск сообществ методом максимизации модулярности.

Дадим более формальное описание алгоритма. Рассматривается граф H с вершинами, соответствующими пользователям U. На первом этапе для каждой пары пользователей $u_i, u_j \in U$ вычисляется количество постов, в которых они оба оставляли комментарии. Это количество присваивается весу ребра в графе w_{ij}. Далее на взвешенном графе H запускается алгоритм максимизации модулярности Louvain [10].

5. Результаты экспериментов

Применение разработанного алгоритма к датасету ВКонтакте дает результаты, представленные в табл. 1 и 2. В первой показаны 10 постов с наибольшей важностью $Reach(p)$. Входящая степень – число уникальных пользователей-комментаторов. Видно, что нет явной корреляции между посчитанной важностью поста и числом лайков, несмотря на начальную инициализацию значений важности. Эксперименты показали, что другие варианты начальной инициализации (число просмотров, все поровну 1) дают такой же или близкий итоговый результат.

Табл. 1: Топ-10 постов по важности в датасете Вконтакте
Table 1. Top-10 important posts in Vkontakte

Пост	**Важность**	**Вход.степень**	**Число лайков**
106351883_1490574	174.34	39	478
33863926_366207	165.98	12	105
33863926_302228	159.97	26	623
106351883_1198977	156.19	33	48
106351883_1354344	155.87	14	25
106351883_1451193	152.16	60	213
176135883_39907	150.87	12	116
39490001_864801	150.16	5	29
150269648_306785	149.06	44	141
150269648_309372	147.62	32	304

Во второй таблице даны 10 наиболее влиятельных пользователей на основании значения $Inf(u)$. Исходящая степень – число прокомментированных пользователем постов. Хотя здесь можно наблюдать корреляцию активности комментирования и влиятельности, что довольно естественно, видно, что ранжирование по ним не совпадает. То есть, пользователь, который

пишет больше постов, не обязательно является более влиятельным. Играет роль также и важность постов, которые он комментирует. Этот результат находится в соответствии с выводами из работ по определению влиятельности пользователей в социальных сетях Twitter и Pinterest [25, 26].

Пользователи в табл. 2 с идентификатором, начинающимся со знака минус, соответствуют владельцам групп, размещавшим посты, то есть они комментировали от имени соответствующей группы.

Табл. 2. Топ-10 пользователей по влиятельности в датасете Вконтакте
Table 2. Top-10 influential users in Vkontakte

Пользователь	**Влиятельность**	**Исх.степень**
-69217442	7952.7	2787
225117304	1102.9	3119
580849174	606.0	1816
161096326	483.1	1122
144232132	320.1	876
616137950	290.3	562
295325982	268.0	1010
-195744122	256.1	589
510446853	250.3	865
572780482	244.9	754

Далее строился граф связей по интересам между пользователями. Для 64441 вершин граф содержит 6 584 434 ребер связей. Метод поиска сообществ находит около 30 сообществ (в зависимости от запуска). Самые большие сообщества имеют размеры 18 059, 12 946, 12 635, 9920, 4752, 1172 вершин. Если проверить, в каких сообществах оказываются топ-100 влиятельных пользователей, то получится, что 70 из них в наибольшем, в следующем по размеру 16, затем 10 и наконец 4 в сообществе с 9920 вершинами. Кроме того, можно посчитать среднюю влиятельность пользователей внутри сообщества. Результаты приведены в табл. 3. Самое большое сообщество оказывается самым влиятельным в среднем. Маленькие сообщества почти всегда содержат маловлиятельных пользователей. Из таблицы наблюдается положительная корреляция между размером сообщества, числом наиболее влиятельных пользователей и его средней влиятельностью. Однако, сообщества размеров 1054 и 618 выбиваются из этой зависимости. Можно предположить, что они образуют отдельные группы со своими лидерами мнений, и потому могут быть интересны для более детального анализа.

Табл. 3. Распределение влиятельных пользователей по сообществам в датасете Вконтакте
Table 3. Distribution of influential users over ciommunities.

Размер сообщества	**Средняя влиятельность**	**Число влиятельных из топ-100**
18 059	3.141	70
12 946	1.505	16
12 635	1.217	10
9920	0.714	4
4752	0.729	0
1172	0.243	0
1088	0.429	0
1054	2.220	0

618	2.799	0
549	0.886	0
394	0.619	0
386	1.212	0
126	0.133	0
97	0.069	0
72	0.158	0
36	0.261	0
17	0.014	0
15	0.000	0
14	0.004	0
12	0.009	0
11	0.016	0
8	0.003	0
7	0.000	0
6	0.000	0
5	0.000	0
3	0.000	0
2	0.000	0
1	0.000	0

Приведем также рзультаты поиска важных видео и влиятельных пользователей для датасета YouTube (табл 4, 5). В целом наблюдается аналогичная картина, что и для соответствующих результатов на датасете ВКонтакте.

Анализ сообществ здесь не проводился, поскольку из-за большого общего числа уникальных пользователей и в то же время большого числа различных комментаторов на одно видео, граф связей по интересам между ними содержит существенное число ребер, что требует существенных вычислительных ресурсов для применения метода Louvain. В таком случае в качестве альтернативы могут быть рассмотрены методы поиска сообществ, имеющие меньшую сложность алгоритма, либо же можно сократить число ребер графа путем введения минимального порога на число общих комментариев для пользователей.

Табл. 4. Топ-10 видео по важности в датасете YouTube
Table 4. Top-10 important videos in YouTube.

Пост	**Важность**	**Вход.степень**	**Число лайков**
95ReakCrKX0	108084	17966	323038
AR6ovvs6Ihg	82268	63466	704654
91yZNamfN1I	73368	21811	293005
vps43rXgaZc	71278	103523	1247927
ttPXXyUyx6Q	69170	17568	255072
QDHYQ9Nd-GU	64301	37334	510176
PWt27h_scaY	61658	13867	164038
602k5_r9ZCs	59179	7389	78470
SgV0-0puqWM	58458	19046	216249
skFNJ3tB67M	57957	12069	134189

Табл. 5. Топ-10 пользователей по влиятельности в датасете YouTube
Table 5. Top-10 influential users in YouTube.

Пользователь	**Влиятельность**	**Исх. степень**
UCMCgOm8GZkHp8zJ6l7_hIuA	765623	22
UCsAw3WynQJMm7tMy093y37A	35067	15
UC-TsVR4jjKT7ZSXoiYQZNIQ	12815	5
UChG-0MFkgZxAwcdxi0mU_JA	12498	2
UC-2rXPfDuLZn2qKmXaR5jWw	11970	1
UCEbCKFjRBtSAa83ROejtyeA	11112	4
UC53JOxGaJlh6vD0ZHUjWhsA	9987	1
UC5RRYPRf06pHSgUOiCy97JA	9391	1
UC7LzUL-OCMf7ehTZn81lXEw	9109	6
UCegPwkRt1G-Y87TEuvyg64w	8664	1

6. Заключение

В рамках НИР проведен обзор методов поиска пользователей социальных сетей. Описаны несколько популярных классов подходов, основанных на информации о графовой структуре сети: методы на основе локальной оптимизации, вероятностные модели, методы распространения меток и методы, основанные на эго-сообществах. Затем дан обзор методов выявления влиятельных пользователей в социальных сетях. Сначала описаны подходы для случая изначально скрытого графа, затем более подробно рассмотрены несколько алгоритмов для случая полной информации о графе. На основе этих методов разработан алгоритм, предложенный для решения поставленной в работе задачи. Далее следует описание неклассических методов поиска сообществ – основанные на деятельности, на основе темы контента и на основе взаимодействия. В конце обзора приводится пример использования текстовой информации сообщений пользователей.

На втором этапе проведены экспериментальные исследования разработанного метода на двух коллекциях данных: из сети ВКонтакте и из YouTube. В обоих датасетах представлены данные о некотором контенте, размещаемом на платформе (посты для ВКонтакте и видео для YouTube) и комментарии пользователей к ним. Модель данных представляет собой двудольный граф, где двумя множествами вершин являются пользователи и контент, а ребра соответствуют комментированию. Итеративный алгоритм вычисляет значение влиятельности для каждого пользователя и величину важности элемента контента. В соответствии с основной идеей алгоритма, пользователь влиятельный, если 1) на его комментарии много отвечают и 2) он комментирует важные элементы контента. Важность элемента контента, в свою очередь, определяется тем, что его 1) много комментируют и 2) комментируют влиятельные пользователи. В итоге самыми влиятельными обычно оказываются наиболее активные пользователями, однако корреляция между влиятельностью и числом комментариев не полная – активный комментатор всего подряд не обязательно будет влиятельным.

После этого были выделены сообщества пользователей, основанные на их общих интересах: чем больше общих элементов контента комментируют два пользователя, тем теснее между ними связь. Алгоритм находит квазиоптимальное разбиение вершин графа на группы, максимизирующее модулярность. По итогам экспериментов наблюдается положительная корреляция между размером сообщества, числом наиболее влиятельных пользователей в нем и средней влиятельностью пользователей сообщества. Некоторые сообщества выбиваются из этой зависимости и потому могут быть интересны для экспертного анализа – возможно, они отражают формирование активных групп по интересам.

Список литературы / References

[1] Conrad Lee, Fergal Reid et al. Detecting highly overlapping community structure by greedy clique expansion. arXiv preprint arXiv:1002.1827, 2010, 10 p.

[2] Andrea Lancichinetti, Filippo Radicchi et al. Finding statistically significant communities in networks. *PloS one*, vol. 6, issue 4, 2011, e18961.

[3] Aaron McDaid and Neil Hurley. Detecting highly overlapping communities with model-based overlapping seed expansion. In Proc. of the International Conference on Advances in Social Networks Analysis and Mining, 2010, pp. 1120-119.

[4] Morten Mørup and Mikkel N Schmidt. Bayesian community detection. Neural computation, vol. 24, issue 9, 2012, pp. 2434-2456.

[5] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In Proc. of the Sixth ACM International Conference on Web Search and Data Mining, 2013, pages 587–596.

[6] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In Proc. of the IEEE 13th international conference on data mining, 2013, pp. 1151-1156.

[7] Mark E.J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. Physical review E, vol. 69, issue 2, 2004, artricle id 026113.

[8] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. Community detection via maximization of modularity and its variants. IEEE Transactions on Computational Social Systems, vol. 1, issue 1, 2014, pp. 46-65.

[9] Nicolas Dugué and Anthony Perez. Directed Louvain: maximizing modularity in directed networks. Research Report hal-01231784, Université d'Orléans. 2015, 15 p.

[10] Vincent D. Blondel, Jean-Loup Guillaume et al. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008, artricle id P10008, 15 p.

[11] Jierui Xie, Boleslaw K. Szymanski, and Xiaoming Liu. SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In Proc. of the IEEE 11th International Conference on Data Mining Workshops, 2011, pp. 344-349.

[12] Zhi-Hao Wu, You-Fang Lin et al. Balanced multi-label propagation for overlapping community detection in social networks. Journal of Computer Science and Technology, vol. 27, issue 3, 2012, pp. 468-479.

[13] Steve Gregory. Finding overlapping communities in networks by label propagation. New journal of Physics, vol. 12, issue 10, 2019, artricle id 103018.

[14] Sucheta Soundarajan and John E Hopcroft. Use of local group information to identify communities in networks. ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 9, issue 3, 2015, pp. 1-27.

[15] Michele Coscia, Giulio Rossetti et al. Demon: a local-first discovery method for overlapping communities. In Proc. of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 615-623.

[16] Bradley S. Rees and Keith B. Gallagher. Egoclustering: overlapping community detection via merged friendship-groups. Lecture Notes in Social Networks, vol. 6, 2013, pp. 1-20.

[17] Nazar Buzun, Anton Korshunov et al. Egolp: Fast and distributed community detection in billion-node social networks. In Proc. of the IEEE International Conference on Data Mining Workshops, 2014, pp. 533-540.

[18] Mohammed Ali Al-Garadi, Kasturi Dewi Varathan et al. Analysis of online social network connections for identification of influential users: Survey and open research issues. ACM Computing Surveys (CSUR), vol. 51, issue 1, 2018, pp. 1-37.

[19] Colin Cooper, Tomasz Radzik, and Yiannis Siantos. A fast algorithm to find all high degree vertices in power law graphs. In Proc. of the 21st International Conference on World Wide Web, 2012, pp. 1007-1016.

[20] Konstantin Avrachenkov, Prithwish Basu et al. Online myopic network covering. UMass Technical Report UM-CS-2012-034, arXiv preprint arXiv:1212.5035, 2012, 18 p.

[21] Konstantin Avrachenkov, Prithwish Basu et al. Pay few, influence most: Online myopic network covering. In Proc. of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2014, pp. 813-818.

[22] Katja Filippova. User Demographics and Language in an Implicit Social Network. In Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 1478–1488.

[23] Konstantin Avrachenkov, Nelly Litvak et al. Quick detection of nodes with large degrees. Internet Mathematics, 10(1-2):1–19, 2014.
[24] 22 Danil Shaikhelislamov, Mikhail Drobyshevskiy et al. Three-step algorithms for detection of high degree nodes in online social networks. In Proc. of the Ivannikov Memorial Workshop (IVMEM), 2020, pp. 43-48.
[25] Daniel M Romero, Wojciech Galuba et al. Influence and passivity in social media. Lecture Notes in Computer Science, vol. 6913, 2011, pp. 18-33.
[26] Zhiguo Zhu, Jingqin Su, and Liping Kong. Measuring influence in online social network based on the user-content bipartite graph. Computers in Human Behavior, vol. 52, 2015, pp. 184-189.
[27] Weishu Hu, Zhiguo Gong et al. Identifying influential user communities on the social network. Enterprise Information Systems, vol. 9, issue 7, 2015, pp. 709-724.
[28] David Darmon, Elisa Omodei, and Joshua Garland. Followers are not enough: A multifaceted approach to community detection in online social networks. PloS one, vol. 10, issue 8, 2015, artricle id e0134860.

Информация об авторах / Information about authors

Роман Константинович ПАСТУХОВ – научный сотрудник. Его научные интересы включают распределенные вычисления, анализ графов, сложные сети, модели случайных графов, анализ социальных сетей, большие данные.

Roman Konstantinivich PASTUKHOV is a researcher. His scope of scientific interests include distibuted computing, graph analysis, complex networks, random graph models, social network analysis, big data.

Михаил Дмитриевич ДРОБЫШЕВСКИЙ – к.ф.-м.н., научный сотрудник. Его научные интересы включают приложения машинного обучения, сложные сети, модели случайных графов, вложения графов, объяснимый искусственный интеллект.

Mikhail Dmitrievich DROBYSHEVSKIY is a Ph.D., researcher. His scope of scientific interests includes machine learning applications, complex networks, random graph models, graph embedding, explainable AI.

Денис Юрьевич ТУРДАКОВ – к.ф.-м.н., заведующий отделом “Информационные системы” ИСП РАН, доцент МГУ. Сфера научных интересов: машинное обучение, интеллектуальный анализ данных, извлечение информации, обработка естественного языка, сложные сети, анализ социальных сетей, большие данные.

Denis Yurievich TURDAKOV – Ph.D., head of the Information Systems Department at ISP RAS, associated professor at MSU. Research interests: machine learning, data mining, information extraction, natural language processing, complex networks, social network analysis, big data.

DOI: 10.15514/ISPRAS-2022-34(5)-9

Исследование методов построения облачных платформенных сервисов и реализаций стандарта TOSCA

[1,2] *А.А. Борисова, ORCID: 0000-0003-4558-7872 <aaborisova_4@edu.hse.ru>*
[2] *О.Д. Борисенко, ORCID: 0000-0001-8297-5861 <al@somestuff.ru>*

[1] *Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20*
[2] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*

Аннотация. В статье рассматриваются и сравниваются различные инструменты автоматизации управления ресурсами в облаке. Изменения в архитектуре программного обеспечения и подходов к разработке требуют автоматизации процессов управления развертывания и дальнейшего сопровождения по в разных средах. В разд. 2 представлен подробный обзор инструментов с примерами конфигураций, а также разбор релевантных статей, рассматривающих различные инструменты автоматизации и эффективность их внедрения. В разд. 3 представлен проект решения по объединению оркестраторов, разработанных в ИСП РАН, для получения инструмента с функционалом, которого нет у конкурентов.

Ключевые слова: оркестрация; развертывание ПО; TOSCA

Для цитирования: Борисова А.А., Борисенко О.Д. Исследование методов построения облачных платформенных сервисов и реализаций стандарта TOSCA. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 143-162. DOI: 10.15514/ISPRAS-2022-34(5)-9

Благодарности: Работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации (соглашение №075-15-2022-294 от 15 апреля 2022)

Research of Construction Methods for Cloud Services and Overview of the Implementations TOSCA Standard

[1,2] *A.A. Borisova, ORCID: 0000-0003-4558-7872 <aaborisova_4@edu.hse.ru>*
[2] *O.D. Borisenko, ORCID: 0000-0001-8297-5861 <al@somestuff.ru>*

[1] *HSE University,*
20, Myasnitskaya st., Moscow, 101000 Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. This paper overview and compares various tools for automating resource management in the cloud. Changes in software architecture and development approaches require automation of deployment management processes and further maintenance of software in different environments. Chapter 2 provides a detailed overview of the tools with sample configurations, as well as a breakdown of relevant articles that look at various automation tools and the effectiveness of their implementation. Chapter 3 presents a draft solution for combining orchestrators developed at ISP RAS to obtain a tool with functionality that competitors do not have.

Keywords: Orchestration; Software Deployment; TOSCA

For citation: Borisova A.A., Borisenko O.D. Research of Construction Methods for Cloud Services and Overview of the Implementations TOSCA Standard. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 143-162 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-9

Acknowledgements. This work was supported by the Ministry of Science and Higher Education of the Russian Federation, agreement No. 075-15-2022-294 dated 15 April 2022.

1. Введение

В индустриальной разработке программного обеспечения наблюдается переход от монолитных архитектур к микросервисным. Микросервисом называют сервис, у которого определена роль и набор процессов, которые он обрабатывает. Преимущество по сравнению с монолитными архитектурами состоит в том, что управление и масштабирование микросервиса может производиться независимо от других микросервисов системы. При этом сервисы можно объединить в группы по специфике использования (например: базы данных, служебные сервисы, сервисы приложений и пр.). Для каждой группы микросервисов жизненный цикл управления – разный. Микросервисы общаются между собой по протоколам передачи данных через программные интерфейсы приложений или передачу сообщений. Архитектура разрабатываемого ПО влияет на то, как будет организованы разработка, поставка и сопровождение ПО, обслуживание инфраструктуры, и организация команды. Переход от монолитной архитектуры к микросервисной влечет за собой переход подходов: обслуживания баз данных; обслуживания инфраструктуры; мониторинга; организации процесса разработки.

Внедрение техник автоматизации процессов разработки (DevOps – development and operations) дает значительный прирост скорости поставки ПО, делает итерации разработки короче и увеличивает частоту поставки. К тому же уменьшается количество отказов системы. Это достигается за счет того, что каждый микросервис поставляется в виде контейнера. Контейнер — это единица поставляемого ПО, содержащая в себе все необходимые для его работы библиотеки и зависимости.

В стандартный жизненный цикл монолитного приложения входит *ручное*:

- развертывание;
- обновление;
- удаление;
- размещение;
- создание резервной копии (англ. backup);
- восстановление из сохраненной копии.

Для управления микросервисным системами "ручной" подход (и даже автоматизация отдельных задач) неприменим, т.к. количество сервисов для одной системы может измеряться тысячами. Поэтому ЖЦ был расширен и автоматизирован. Для микросервисов продвинутый ЖЦ позволяет:

- автоматическое управление стандартным ЖЦ ПО;
- автоматическое масштабирование;
- автоматический возврат ПО к предыдущей версии (или "откат изменений" - англ. rollback);
- развертывание изменений для части сервисов, а не на все сразу (canary и blue-green развертывания).

К тому же, были разработаны подходы автоматического управления, которые способны реагировать на изменения в микросервисах и по необходимости менять шаг жизненного цикла (напр. автоматическое восстановление при возникновении ошибок во время развертывания). Для эффективного управления сложными системами необходимо повышать прозрачность и просматриваемость (observability) системы. Необходимо отслеживать очаги

аномальной работы, падений (иногда отказ одного сервиса тянет за собой отказ связанных сервисов). Все ошибки записываются, отслеживаются и измеряются метриками - этот процесс называется мониторингом системы. Также вводится понятие состояние сервиса и предоставляются хранилища состояний, для отслеживания состояния сервиса и системы в целом в конкретный период времени. Примеры состояний: запущен, в ожидании, прерван. Не всегда отображается действительное состояние сервиса, он может быть запущен, но может не отвечать на определенного рода запросы. Для более детальной информации используются обработчики событий, которые посылают запросы на проверку "здоровья" (работоспособности) сервиса. Все это дает четкое представление о работе системы. Как итог, видны критические сервисы, и сервисы, на которые идет большая/меньшая нагрузка. Все это позволяет сделать систему в целом более управляемой, позволяет быстрее реагировать на изменение потребления ресурсов системы (напр. автоматическое масштабирование при высоком потреблении ресурсов) [3].

Стоит отметить, что для работы распределенной системы с микросервисной архитектурой, необходимо поддерживать рабочее состояние сети и обеспечивать связывание с другими микросервисами и устаревшими системами по сети (используя различные протоколы и сервисы обмена сообщениями).

Рассмотрим все вышеописанное на примере. Предположим, что для продукта была выбрана микросервисная архитектура. При этом происходят следующие изменения: команда продукта делится на множество мелких подкоманд, которые отвечают каждая за свой сервис. Развертывание микросервисов может происходить независимо друг от друга и с частотой менее одного дня. Само развертывание микросервиса длится не долго. Управление микросервисом происходит автоматически. Микросервис можно развернуть на тестовой среде и протестировать изменения перед развертыванием в промышленную среду. При этом для микросервисов доступно продвинутое управление ЖЦ ПО. Устройство системы позволяет быстро реагировать на изменения рынка (разрабатывать и поставлять обновления ПО как можно быстрее), и получать на выходе успешный и конкурентоспособный продукт.

Однако реализация перехода с монолитных на микросервисные архитектуры оказалась проблематичной в направлениях создания инфраструктуры, а также развертывания и управления сразу множеством сервисов. Благодаря появлению контейнеров и инструментов оркестрации проблемы были решены.

Для конфигурации и управления монолитным ПО используются инструменты автоматизации стандартных задач ЖЦ ПО. Но для микросервисов этого недостаточно, поэтому используются инструменты оркестрации, которые позволяют автоматизировать управление набором отдельных автоматизированных задач. Термин «оркестрация» не имеет зафиксированного определения и границы сильно размыты. Однако можно описать набор задач, которые должен решать оркестратор. А именно: автоматизировать набор задач по развертыванию, координации и управлению инфраструктурой, ПО и сервисами. Процесс оркестрации зависит от того, как вообще организована поставка ПО для конкретного продукта. Включение оркестрации в процессы позволяет ощутимо для компании и не ощутимо для пользователя поставлять обновления, менять схему взаимодействия сервисов, исправлять перебои и пр. Гибкая разработка ПО напрямую связана с гибким управлением и поставкой. Это позволяет наиболее эффективно использовать ресурсы. С появлением оркестрации появилась возможность делать безопасные Canary, и Blue-Green релизы. Производить частичное тестирование новых функций ПО до того, как масштабировать изменения на всех пользователей. Платформы оркестрации позволяют осуществлять более надежную и стабильную работу системы в целом, а также осуществлять поиск возможных ошибок на ранних этапах и быстрое обслуживание системы, без увеличения количества сотрудников.

Контейнеры более легковесные и управляемые, ресурсы контейнера изолированы от других частей системы. Контейнерам не нужно настраивать и развертывать уровень

инфраструктуры, в нем уже содержится инфраструктура в виде слоя базового образа. Инфраструктурой называют часть сервисного ПО, которая необходима для работы разрабатываемого ПО. Процесс оркестрации контейнеров и виртуальных машин отличается по этапам и построению. Для контейнеров используется платформы оркестрации Docker Swarm и Kubernetes. На начальном этапе пользователь уже имеет «собранный» вручную контейнер, обслуживаемый контейнерным движком.

Настройки сборки прописываются при запуске сборки или конфигурируются на следующем этапе. Далее пользователь пишет конфигурацию сервиса (или группы сервисов). В Kubernetes есть API конфигурации, который абстрактно описывает элементы инфраструктуры. По API для контейнеров можно понять, какую роль выполняет контейнер (напр. рабочая нагрузка, БД, балансировщик и т.д.). Также в конфигурации описано поведение контейнера при отказах, заданы связи с другими контейнерами. Для более сложных функций работы контейнеров друг с другом используются специальные настраиваемые элементы - «операторы».

В отличие от контейнеров, ПО не имеет под собой инфраструктуры. Создание инфраструктуры выполняется не линейно. Перед запуском сценария развертывания может выполниться подготовительный шаг, а после - недостающие параметры подставляются в конфигурацию и запускается скрипт донастройки. Это объясняется тем, что виртуализация на базе гипервизора требует развертывания на аппаратной части, в то время как виртуализация контейнеров выполняет развертывание на уровне операционной системы. Таким образом, при развертывании виртуальных машин на базе гипервизора необходимо развернуть уровень инфраструктуры и настроить его (этот этап называют Infrastructure Provisioning), затем собрать и развернуть на инфраструктуре сервисы (этот этап называют Configuration management). Для каждого уровня свои сценарии развертывания. Связи и отношения между сервисами также заданы в сценарии развертывания (поэтому это не так прозрачно, как у контейнеров). Инструменты конфигурации очень гибкие и работают на высоком уровне абстракции, поэтому никак не ограничивают роль, которую может выполнять сервис, что делает процесс настройки намного более сложным, нежели настройка контейнеров. При наличии таких сценариев можно начинать использовать Оркестратор, который позволит создавать сценарии управления группой сервисов. Далее подключаются сервисы мониторинга и переход на этап обслуживания и мониторинга.

На практике распределенные приложения требуют решений и для обслуживания виртуальных машин, и для обслуживания контейнеров. Для контейнеров стандартом стал Kubernetes – как наиболее мощная и развивающаяся платформа. А для виртуальных машин оркестрация остается сложно реализуемой, потому что уровень виртуализации ближе к аппаратной части. Поэтому оркестраторы поставляются для каждого провайдера по отдельности, либо процесс оркестрации реализуется инструментами конфигурации в полуавтоматическом режиме. Задача построения мультиоблачного оркестратора с поддержкой нескольких инструментов конфигурации остается актуальной.

На пути реализации такого оркестратора был создан стандарт TOSCA [2], позволяющий выйти на необходимый уровень абстракции. Стандартом описаны сущности облачных вычислений без привязки к инструменту или провайдеру. На базе стандарта стало возможно реализовать мультиоблачный оркестратор.

Еще одной актуальной проблемой является измерение эффективности внедрения оркестратора.

2. Обзор решений

В обзоре технологий рассмотрены различные инструменты оркестрации. Сравнение инструментов возможно, при учете следующих факторов: уровень виртуализации, уровень доступа к ресурсам, поддержка одного/нескольких облачных провайдеров, возможность

взаимодействия с инструментами конфигурации, графический интерфейс построения топологий и пр. Эти факторы необходимо учитывать, т.к. некорректно сравнивать оркестраторы различных уровней доступа к ресурсам или же сравнивать производительность инструмента конфигурации и оркестратора контейнеров. Поэтому технологии разбиты на 4 категории, в каждой из которых описан обзор и предоставлено сравнение инструментов.

2.1 Инструменты, позволяющие осуществлять оркестрацию

2.1.1 Terraform

Terraform – инструмент, осуществляющий оркестрацию для различных облачных провайдеров (более 30) уровня IaaS на этапе настройки и развертывания [4]. Для некоторых провайдеров доступен PaaS уровень. Поддерживается создание, удаление и обновление единиц развертывания. Единицей развертывания является модуль, поддерживаемый для данного провайдера. Конфигурация задается через Terraform configuration – шаблон описания конфигурации для каждого провайдера на языке Terraform. Не поддерживаются функции масштабирования, мониторинга, обслуживания ЖЦ ПО (например, автоматическое восстановление в случае отказа).

Terraform предоставляет часть функций бесплатно. Функции Terraform: удаленное хранилище состояний, возможность удаленных запусков, соединение с инструментами контроля версий (Github, Gitlab, Bitbucket, Azure Devops). В индустриальной версии Terraform есть возможность поддержки собственного облачного сервиса Terraform. При этом облако можно разделить между организациями, у которых будет отдельный счет для оплаты ресурсов и собственное рабочее пространство. Внутри каждой организации ресурсы можно разделить и ограничить между командами разработки и пользователями. Terraform можно самостоятельно расширять, добавляя поддержку необходимого провайдера. Однако официальные модули крупных облачных провайдеров обновляются со значительной задержкой. Инструмент позволяет описывать зависимости между модулями. Но при развертывании и удалении проверяется только порядок, а не совместимость и наличие циклов.

Terraform подходит для простых задач по подготовке инфраструктуры для различных облачных провайдеров. На листинге 1 представлен пример сценария развертывания с использованием Terraform. Оркестрация сервисов PaaS уровня менее полезна, потому что инструмент не поддерживает продвинутых функций оркестрации. Инструмент не предоставляет доступ к своим собственным PaaS сервисам, а лишь управляет сервисами провайдеров. К тому же язык Terraform требует изучения и не позволяет использовать внутри себя вставки с вызовом кода. Terraform позволяет описать топологию, однако шаблоны описания не универсальны и для каждого провайдера необходимо создавать сценарий.

Terraform не предоставляет ресурсы по запросу, но позволяет управлять этими ресурсами.

```
#### INSTANCE DB ####

# Create instance
#
resource "openstack_compute_instance_v2" "db" {
  name        = "front01"
  image_name  = var.image
  flavor_name = var.flavor_db
  key_pair    = openstack_compute_keypair_v2.user_key.name
  user_data   = file("scripts/first-boot.sh")
  network {
    port = openstack_networking_port_v2.db.id
```

```
  }
}

# Create network port
resource "openstack_networking_port_v2" "db" {
  name           = "port-instance-db"
  network_id     = openstack_networking_network_v2.generic.id
  admin_state_up = true
  security_group_ids = [
    openstack_compute_secgroup_v2.ssh.id,
    openstack_compute_secgroup_v2.db.id,
  ]
  fixed_ip {
    subnet_id = openstack_networking_subnet_v2.http.id
  }
}

# Create floating ip
resource "openstack_networking_floatingip_v2" "db" {
  pool = var.external_network
}

# Attach floating ip to instance
resource "openstack_compute_floatingip_associate_v2" "db" {
  floating_ip = openstack_networking_floatingip_v2.db.address
  instance_id = openstack_compute_instance_v2.db.id
}

#### VOLUME MANAGEMENT ####

# Create volume
resource "openstack_blockstorage_volume_v2" "db" {
  name = "volume-db"
  size = var.volume_db
}

# Attach volume to instance instance db
resource "openstack_compute_volume_attach_v2" "db" {
  instance_id = openstack_compute_instance_v2.db.id
  volume_id   = openstack_blockstorage_volume_v2.db.id
}
```

Листинг 1. Пример сценария развертывания ресурсов Openstack в Terraform
Listing 1. Example of a Terraform deployment scenario for OpenStack

2.1.2 Configuration management software

Инструменты, такие как Ansible, Chef, Puppet, имеют схожий функционал [5, 6, 7].

Все они поддерживают несколько облачных провайдеров, для которых описаны модули, с которыми они работают. В сценариях можно описывать не только развертывание, но и любую конфигурацию. В сценарии можно описать развертывания различных модулей различных провайдеров. Сценарии не являются шаблонами, не предназначены для пере

использования между провайдерами. Инструментами конфигурации поддерживаются все уровни доступа к ресурсам. Инструменты конфигурации позволяют осуществлять процессы оркестрации, однако написание вручную шаблонов и вызов функций не дают им преимущество по сравнению с оркестраторами.

Рассмотрим один из инструментов конфигурации.

Ansible

Ansible – мощный инструмент настройки для автоматизации управления вычислительными ресурсами в облаке. Благодаря декларативному описанию на предметно-ориентированном языке в формате YAML специалисты могут легко настраивать, развертывать и контролировать инфраструктуры, приложения, сети и множество других сущностей. Файлы описания конфигурации называются Playbook, и они состоят из списка задач для исполнения, будем называть их «сценариями Ansible». На листинге 2 представлен пример сценария развертывания с использованием Ansible.

Однако использование этого инструмента требует понимания правил построения Ansible сценариев, а также знания точного API провайдера конкретного облачного провайдера. Это вызывает проблемы при использовании, когда конфигурация приложения уже выполнена, и требуется развернуть приложение с использованием услуг другого облачного провайдера. Это то, что отличает его от шаблонов унифицированного абстрактного описания, которые не зависят от конкретной технологии или поставщика услуг.

```
- name: Create OpenStack component openstack cluster
  hosts: localhost
  tasks:
  - name: Create OpenStack component server
    os_server:
      config_drive: false
      name: server_master
      flavor: '{{ id_3387 }}'
      image: '{{ name_8282 }}'
    register: server_master_server
  - set_fact:
      server_master_server_list: '{{ server_master_server_list
      | default([])
        }} + [ "{{ item.id }}" ]'
    loop: '{{ server_master_server.results | flatten(levels=1)  }}'
    when: item.id  is defined
  - set_fact:
      server_master_server_list:
        server_master_server_ids: '{{ server_master_server_list }}'
    when: server_master_server_list is defined
  - lineinfile:
      path: '{{ playbook_dir }}/id_vars_example.yaml'
      line: 'server_master_server_delete: {{ server_master_server.id }}'
    when: server_master_server.id is defined
  - lineinfile:
      path: '{{ playbook_dir }}/id_vars_example.yaml'
      line: '{{ server_master_server_list | to_nice_yaml }}'
    when: server_master_server_list is defined
  - fail:
```

```
      msg: Variable server_master_server is undefined! So it will not be
deleted
    when:      server_master_server_list      is      undefined      and
server_master_server.id
      is undefined
    ignore_errors: true
```

Листинг 2. Пример сценария развертывания ресурсов Openstack в Ansible
Listing 2. Example of a Ansible deployment scenario for OpenStack

Ansible Galaxy представляет собой цифровую библиотеку открытых исходных кодов, которая содержит более 27 тысяч сценариев конфигурации сервисов, оформленных в виде ролей Ansible [8]. Далеко не все они работают, и большая часть из них требует дополнительной подготовки перед использованием.

Ansible используется множеством компаний для сопровождения их сервисов, а Galaxy стал местом для обмена лучшими практиками. Число людей, входящих в это сообщество, превосходит 299 тысяч человек.

2.2 Оркестраторы с поддержкой одного облачного провайдера

2.2.1 Cloud Formation

Cloud Formation – оркестратор облачного провайдера Amazon AWS уровня IaaS [9]. Поддерживает создание, удаление и обновление единиц развертывания. Единицей развертывания является Stack – набор ресурсов (напр. балансировщик, виртуальные машины, хранилища данных). Конфигурация задается через AWS templates в формате YAML/JSON на языке Cloud Formation template language. Шаблоны можно проверять на корректность. Можно задавать связи типа «Depends on» внутри шаблона. Поддерживается функция управления ЖЦ ПО сине-зеленое развертывание, сохранение резервной копии и восстановление из нее, автоматическое масштабирование. Пример сценария развертывания приведен на листинге 3.

Также, по шаблону можно создать "безсервисную" конфигурацию и получить развертывание уровня FaaS (функция как сервис). Шаблоны покрывают все ресурсов провайдера, в том числе сервисы мониторинга и сопровождения ЖЦ ПО. Провайдер Amazon имеет встроенный инструмент графического задания топологий формата CloudFormation. CloudFormation это полноценный оркестратор, покрывающий все функции для одного облачного провайдера Amazon AWS.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description"    :    "AWS    CloudFormation    Sample    Template
EC2InstanceWithSecurityGroupSample: Create an Amazon EC2 instance
running the Amazon Linux AMI. The AMI is chosen based on the region in
which the stack is run. This example creates an EC2 security group for
the instance to give you SSH access. **WARNING** This template creates
an Amazon EC2 instance. You will be billed for the AWS resources used if
you create a stack from this template.",

  "Parameters" : {
    "KeyName": {
      "Description" : "Name of an existing EC2 KeyPair to enable
       SSH access to the instance",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription" : "must be the name of an existing
       EC2 KeyPair."
```

```
    },

    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t1.micro", "t2.nano", "t2.micro", "t2.small",
"t2.medium",    "t2.large",    "m1.small",    "m1.medium",    "m1.large",
"m1.xlarge",   "m2.xlarge",   "m2.2xlarge",   "m2.4xlarge",   "m3.medium",
"m3.large",    "m3.xlarge",    "m3.2xlarge",    "m4.large",    "m4.xlarge",
"m4.2xlarge",  "m4.4xlarge",  "m4.10xlarge",  "c1.medium",  "c1.xlarge",
"c3.large",   "c3.xlarge",   "c3.2xlarge",   "c3.4xlarge",   "c3.8xlarge",
"c4.large",   "c4.xlarge",   "c4.2xlarge",   "c4.4xlarge",   "c4.8xlarge",
"g2.2xlarge",   "g2.8xlarge",   "r3.large",   "r3.xlarge",   "r3.2xlarge",
"r3.4xlarge",  "r3.8xlarge",  "i2.xlarge",  "i2.2xlarge",  "i2.4xlarge",
"i2.8xlarge",  "d2.xlarge",  "d2.2xlarge",  "d2.4xlarge",  "d2.8xlarge",
"hi1.4xlarge",     "hs1.8xlarge",     "cr1.8xlarge",     "cc2.8xlarge",
"cg1.4xlarge"]
,
      "ConstraintDescription" : "must be a valid EC2 instance type."
    },

    "SSHLocation" : {
      "Description" : "The IP address range that can be used to SSH to
the EC2 instances",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern":
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form
x.x.x.x/x."
   }
 }
```

Листинг 3. Пример сценария развертывания ресурсов Amazon в CloudFormation
Listing 3. Example of a CloudFormation deployment scenario for Amazon

2.2.2 Heat Orchestrator

Heat – инструмент, осуществляющий оркестрацию для облачного провайдера OpenStack уровня IaaS [10]. Поддерживает создание, удаление и обновление единиц развертывания. Единицей развертывания является Stack – набор ресурсов (напр. балансировщик нагрузки, виртуальные машины, хранилища данных). Конфигурация задается через Heat Orchastration Templates – шаблоны TOSCA ненормативных типов Openstack. Пример сценария развертывания показан на листинге 4.

Продвинутые функции управления жизненным циклом ПО не поддерживаются. Имеется Amazon AWS Query API для поддержки создания сопоставимых ресурсов Cloud Formation. Провайдер OpenStack имеет встроенный инструмент графического задания топологий формата Heat.

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance
```

```
parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
```

Рис. 4. Пример сценария развертывания ресурсов OpenStack в Heat Orchestrator
Fig. 4. Example of a Heat deployment scenario for Openstack

2.2.3 Michman

Michman (разработка ИСП РАН) - это оркестратор уровня PaaS провайдера OpenStack [11, 12]. Отличие оркестратора в том, что сервисы предоставляются по запросу и уровень инфраструктуры настраивается автоматически. Сейчас реализована поддержка следующих сервисов: Apache Spark, Apache Hadoop, Apache Ignite, Apache Cassandra, ClickHouse, CouchDB, CVAT, ElasticSearch with OpenDistro tools, Jupyter, Jupyterhub, Kubernetes, Nextcloud, NFS-Server, Slurm, PostgreSQL, Redis. Есть возможность добавления поддержки собственного сервиса. Michman поддерживает создание и удаление единиц развертывания, а также масштабирование сервиса. Единицей развертывания является сервис, который пользователь выбирает и настраивает через графический или программный интерфейс. Michman использует Ansible, как инструмент конфигурации. Для развёртывания некоторого сервиса пользователю необходимо задать выбрать сервис, а затем выполнить одну команду для развертывания. Michman автоматически подберет параметры IaaS и обработает зависимости между сервисами. Результатом программы является сервис, развернутый в облаке Openstack. Не поддерживаются функции, мониторинга и продвинутого обслуживания ЖЦ ПО (например, автоматическое восстановление в случае отказа). Нет графического редактора топологий.

2.3 Оркестраторы нескольких облачных провайдеров

2.3.1 Cloudify

Cloudify – мультиоблачный оркестратор уровня IaaS и контейнеров .[13]. Расширяем при помощи добавления плагинов. На данный момент плагины реализованы для интеграции с

инструментами конфигураций, развертывания IaaS ресурсов и осуществления оркестрации. Cloudify поддерживает создание, удаление и обновление единиц развертывания а также любые операции, которые поддерживают инструменты Terraform, Ansible, Helm (Kubernetes), Docker. Единицей развертывания является Blueprint template – топология, описанная на похожем на стандарт TOSCA предметно-ориентированном языке в формате YAML файла. Пример сценария развертывания приведен на листинге 5.

Шаблоны покрывают ресурсы и операции ЖЦ ПО, реализованные в соответствующих плагинах. Поддерживает интеграцию с CI/CD инструментами Jenkins, Github Actions, CircleCI. Cloudify имеет графический инструмент создания топологий, что делает порог вхождения в Cloudify dsl ниже и позволяет визуально отобразить топологию и связи между элементами.

```
tosca_definitions_version: cloudify_dsl_1_3

imports:
  - https://cloudify.co/spec/cloudify/6.3.0/types.yaml
  - plugin:cloudify-openstack-plugin?version= >=3.2.2
  - plugin:cloudify-ansible-plugin
  - plugin:cloudify-utilities-plugin?version= >=1.22.1
  - includes/hello-world-ansible.yaml

inputs:
  <parameters_here>

dsl_definitions:

  openstack_config: &openstack_config
    auth_url: { get_secret: openstack_auth_url }
    region_name: { get_input: region }
    project_name: { get_secret: openstack_tenant_name }
    username: { get_secret: openstack_username }
    password: { get_secret: openstack_password }
    user_domain_name: { get_input: user_domain_name }
    project_domain_name: { get_input: project_domain_name }

node_templates:

  vm:
    type: cloudify.nodes.openstack.Server
    properties:
      client_config: *openstack_config
      agent_config:
        install_method: none
        key: { get_attribute: [agent_key, private_key_export] }
        user: { get_input: agent_user }
      resource_config:
        name: vm
        image_id: { get_input: image }
        flavor_id: { get_input: flavor }
        user_data: { get_attribute: [ cloud_init, cloud_config ] }
      use_public_ip: true
```

```
    relationships:
      - type: cloudify.relationships.openstack.server_connected_to_port
        target: port
      - type: cloudify.relationships.depends_on
        target: cloud_init
```

Рис. 5. Пример сценария развертывания ресурсов Openstack в Cloudify
Fig. 5. Example of a Cloudify deployment scenario for Openstack

2.3.2 xOpera Orchestrator

xOpera – мультиоблачный оркестратор уровня IaaS провайдера Openstack, FaaS провайдеров Amazon, Azure, Google Cloud Platform и SaaS [14]. В xOpera реализована поддержка соединения между облаками (запрос можно последовательно обрабатывать на ресурсах разных провайдеров). xOpera поддерживает создание, удаление единиц развертывания (обновление происходит как удаление и развертывание новых). Единицей развертывания является TOSCA template – топология, описанная по стандарту TOSCA в формате YAML файла. Шаблоны содержат описания в ненормативных типах, зависящих от провайдера. На каждый ресурс необходимо предоставить артефакты со сценариями создания и удаления на Ansible. xOpera поддерживает интеграцию только с инструментом конфигурации Ansible. Параметры из шаблона TOSCA подставляются в сценарий при запуске оркестратором сценария развертывания Ansible. И шаблоны, и сценарии остаются провайдер-зависимыми. Нельзя добавлять поддержку нового инструмента конфигурации, но можно описать свои ненормативные типы ля добавления нового провайдера.

2.3.3 Clouni

Clouni (разработка ИСП РАН) - это TOSCA мультиоблачный оркестратор уровня IaaS провайдеров OpenStack, Amazon, Kubernetes [15]. Основная особенность этого инструмента - использование нормативных типов TOSCA для их автоматического преобразования в сценарии для поддерживаемых облачных провайдеров. Clouni поддерживает создание и удаление единиц развертывания. Единицей развертывания является TOSCA template – топология, описанная по стандарту TOSCA в формате YAML файла. Инструменты конфигурации для разных провайдеров могут отличаться. Например, для Kubernetes провайдеров используются Kubernetes манифесты, для OpenStack провайдеров - сценарии Ansible. Для развёртывания некоторой инфраструктуры пользователю необходимо описать эту инфраструктуру в TOSCA шаблоне с использованием нормативных типов (параметры, описанные в стандарте TOSCA, которые интуитивно понятны, например, ЦП, размер памяти и т. д.). А затем выполнить одну команду для создания сценария развёртывания. Результатом программы является готовый к развертыванию сценарий или манифест. Еще одна особенность заключается в том, что любой может добавлять поддержку облачных провайдеров, создав определение ненормативных TOSCA типов ресурсов облачного провайдера и схему сопоставления с нормативными типами.

В настоящее время Clouni реализовано несколько нормативных типов TOSCA. А именно преобразование Compute, Network и Port в Ansible Playbooks для провайдеров OpenStack и только Compute в Amazon и в Kubernetes Manifest для контейнерной платформы Kubernetes. Сценарии Ansible генерируются для создания и удаления инфраструктуры. Манифесты Kubernetes следует использовать с командами kubectl apply/delete.

2.3.4 Alien4Cloud

Alien4Cloud - это инструмент моделирования TOSCA с открытым исходным кодом [16]. Его основное отличие состоит в том, что он позволяет разворачивать сервисные кластеры из веб-интерфейса за счёт подключаемых плагинов. Каждый плагин трансформирует запросы и

ответы Alien4Cloud и некоторого TOSCA сервиса, что позволяет использовать внешний TOSCA сервис из веб-интерфейса Alien4Cloud. Кроме того, инструмент предоставляет возможность добавлять собственные расширяющие плагины с определенной логикой (например, дополнительный оркестратор TOSCA). Alien4cloud не меняет вводимые данные и позволяет использовать собственный нормативный тип TOSCA. Реализована функция мониторинга.

Yorc оркестратор - официальный мультиоблачный оркестратор TOSCA для Alien4cloud уровня IaaS провайдеров Amazon, OpenStack, Google Cloud Platform, Kubernetes [17]. Также, Yorc предоставляет Slurm как сервис (PaaS уровень). Yorc использует Alien4cloud типы TOSCA на основе TOSCA Simple YAML Profile v.1.2. Надлежащая документация по обоим инструментам четко описывает, как использовать инструменты вместе. Единицей развертывания является архив, содержащий TOSCA template. Продвинутые функции управления жизненным циклом ПО не поддерживаются. К сожалению, примеры процессов оркестрации из документации нельзя оценить, потому что образцы не обновлялись долгое время и не работали. Однако же удалось подключить Alien4cloud к Openstack и запустить развертывание виртуальной машины, которое оказалось не успешным. По анализу логов, можно убедиться, что Alien4cloud использует Terraform как инструмент развертывание и подставляет готовые сценарии развертывание для заданных ненормативных типов.

2.4 Оркестраторы контейнеров

2.4.1 Docker Swarm

Swarm это специальный режим работы сервиса Docker. Swarm не предоставляет ресурсы по запросу, он только управляет контейнерами. Поддерживает развертывание, удаление, обновление и масштабирование ресурсов. Единицей развертывания является стек - набор сервисов-контейнеров. Конфигурация развертывания задается через конфигурационный файл docker compose. Swarm использует декларативный подход и одновременно выступает в роли инструмента оркестрации контейнеров. Мониторинг сервисов не производится, но можно получить состояние сервиса по запросу. При обновлении стека можно указать порядок обновления сервисов, количество контейнеров, которые можно обновлять одновременно, указать стратегию в случае сбоя (есть функция отката обновлений в случае сбоя). Функции Swarm можно расширять при помощи подключения плагинов. Графический интерфейс не поддерживается.

2.4.2 Kubernetes

Наиболее функциональной платформой оркестрации контейнеров на данный момент является Kubernetes. Kubernetes выступает как провайдер, позволяя предоставлять ресурсы на всех уровнях IaaS, PaaS, SaaS [18]. А также как платформа оркестрации, которая управляет ресурсами. Поддерживает все функции обслуживания ЖЦ ПО, в том числе масштабирование (как горизонтальное, так и вертикальное), а также мониторинг ресурсов и сервисов. Конфигурация развертывания задается через Kubernetes manifest. Kubernetes использует декларативный подход и одновременно выступает в роли инструмента конфигурации и оркестрации. Шаг со сборкой сервиса не требуется (т.к. оркестратор оперирует контейнерами, в которых уже содержаться собранный библиотеки и зависимости). Поэтому Kubernetes позволяет настроить конфигурацию работы между сервисами и полностью описать работу сервиса в различных случаях, например в различных состояниях. В каждый момент времени Kubernetes отслеживает состояние сервиса и принимает действие, например, в случае отказа. Состояния отслеживаются при помощи механизма "проверки работоспособности" сервиса (англ. Healthcheck). Kubernetes Можно интегрировать с

различными инструментами конфигурации, интеграции и поставки ПО. Минусом является отсутствие программ для графического редактирования топологий.

2.5 Сравнение Cloudify и Terraform

В статье [19] сравнивают производительность развертывания двух инструментов – Terraform (инструмент автоматизации) и Cloudify (оркестратор на базе стандарта TOSCA), а также рассматривают функции оркестраторов: OpenStack Heat, CloudFormation, и Cloud Assembly. Многие инструменты описаны поверхностно, авторы дают представление о функциональности, но не позволяют сделать сравнительный анализ и выявить сильные и слабые стороны инструментов.

Производительность сравнивают только для Cloudify и Terraform. Для эксперимента авторы предложили развернуть инфраструктуру (IaaS) и приложение Wordpress на ней (PaaS), т. к. Wordpress популярный вариант приложения. Было решено развернуть облачное приложение у 3-х различных провайдеров (Aws, Google cloud platform and Azure) под это требование из рассмотренных в статье оркестраторов подходят только Cloudify и Terraform.

На хост с оркестраторами были установлены сервисы мониторинга, которые следили за состоянием во время развертывания и удаления инфраструктуры. Авторы статьи использовали похожие конфигурации провайдеров. Однако, стоит заметить, что Terraform не оркестратор, и в данном случае выступает, как инструмент конфигурации. В то время как Cloudify - оркестратор, главным преимуществом которого является не работа "на скорость", а работа с несколькими облачными провайдерами и инструментами конфигурации. То есть само по себе сопоставление не является благоразумным. Авторы выбрали Ansible, как инструмент конфигурации для эксперимента, но не упомянули об этом. Стоит заметить, что Cloudify может работать и с Terraform и причины выбора Ansible авторы не называют. То есть в случае с Cloudify, неявно используется инструмент конфигурации, а Terraform работает напрямую с провайдером и очевидно, что результаты будут в худшую сторону для Cloudify. Эти результаты подтверждены графиками, но не отражают действительных преимуществ оркестратора. В заключении авторы делают упор на высокую производительность инструмента конфигурации Terraform и значительные преимущества в производительности. Однако эксперимент был проведен только в части развертывания и удаления сервисов. У Cloudify есть значительные преимущества - унифицированные шаблоны топологий, графический интерфейс для отображения топологии, возможность полномасштабной оркестрации (Terraform осуществляет оркестрацию только на IaaS уровне).

Авторы дают ссылку на шаблоны развертывания, которые они используют. По ним видно, какую проблему пытается решить стандарт TOSCA.

1) Оба инструмента развертывают одну и ту же инфраструктуру.
2) Для использования Terraform необходимо выучить синтаксис языка инструмента, а для использования Cloudify с несколькими провайдерами и инструментами конфигурации - достаточно научиться использовать только TOSCA.

С другой стороны, Cloudify использует свою интерпретацию стандарта TOSCA - TOSCA DSL. И поддерживает только ненормативные типы, которые добавляются в виде плагинов, что лишает преимуществ инструмент по сравнению с другими TOSCA Оркестраторами.

2.6 ToolKit Deployment Manager

В статье [20] рассматривается инструмент автоматизации развертывания (оркестратор) для различных дистрибутивов облачного провайдера Openstack на уровне IaaS.

Реализованный оркестратор должен отвечать требованиям по координации, автоматизации, подготовке и мониторингу ресурсов. Инструмент называется "ToolKit Deployment Manager" и состоит из набора блоков-контроллеров, которые отвечают за управлением (развертыванием и удалением): Сетью, Платформой, ПО, Сервисами.

Инструмент был реализован и проведено исследование эффективности оркестратора, которое наглядно показывает пользу. Процесс оркестрации проводился при помощи сценариев Развертывания Ansible. В качестве метрик для измерения возможности развертывания взяты:

- "среднее время развертывания" – время от вызова скрипта развертывания до перевода состояния как "запущено";
- "надежность развертывания" – отношение успешных развертываний к общему количеству запусков;
- "количество шагов развертывания" – общее количество шагов, требующееся для выполнения развертывания;
- "параметры, введенные во время развертывания" – столько раз пользователю необходимо вручную ввести параметры во время выполнения развертывания;
- "сумма шагов развертывания" – сумма количества шагов развертывания и количества вмешательств пользователя.
- "возможность развертывания" – итоговое количество усилий для успешного развертывания ((1-"Надежность развертывания") + 1) * "Сумма шагов развертывания".

Эти метрики действительно показывают эффективность внедрения инструментов оркестрации, т.к. они учитывают время, которое пользователь бы затратил без использования оркестратора. Также, необходимо добавить метрики, которые показывают количество артефактов развертывания, с которыми приходится работать. Т.к. версионирование, учет и написание новых сценариев занимает большое количество времени, а оркестратор помогает автоматизировать эти задачи и уменьшить количество сценариев.

Результаты, полученные в статье, демонстрируют сокращение количества ошибок развертывания и потраченного времени, что говорит об эффективном внедрении оркестратора.

3 Исследование и построение решения задачи

3.1 Сравнительный анализ методов построения облачных платформенных сервисов

Построение сервисов для виртуализации на базе гипервизора и на базе ОС (контейнеры) отличается. Для развертывания контейнеров не требуется развертывание уровня инфраструктуры. Kubernetes фактически стал стандартом в оркестрации и поставке сервисов уровня PaaS для контейнеров. Вопрос построения сервисов, развернутых на виртуальных машинах, остается открытым. Это связано с тем, что только провайдеры решают, предоставлять открытые интерфейсы для управления ресурсами в облаке или оставлять оркестрацию за собой. В виду этого, модули, которые поддерживаются инструментами конфигурации ограничены в возможностях. Например, ни Terraform, ни Ansible не могут управлять ресурсами в облаке Amazon также гибко, как и Cloud Formation. Именно поэтому, новейшие оркестраторы используют унифицированное описание топологий по стандарту TOSCA, которое помогает абстрактно описывать ресурсы и ЖЦ ПО, не привязываясь к реализациям провайдера. Кроме того, использование стандарта помогает настроить сообщение и работу между облачными провайдерами, а это безусловное преимущество TOSCA-оркестраторов.

Для развертывания виртуальных машин в облаке активно используются инструменты конфигурации, а также Terraform, однако это инструменты конфигурации можно считать устаревшими, ведь они не отвечают уровню, который необходим для оркестрации и масштабирования сложных микросервисных систем.

В табл. 1 видно, что оркестраторы в основном не предоставляют ресурсы, а лишь управляют ими (ведь это задача провайдера). Однако, на самом деле возможно построение такого оркестратора, который бы предоставлял платформенные сервисы по запросу. Примером реализации является Michman. Сложность построения кроется в том, что Michman "под капотом" развертывает и настраивает уровень инфраструктуры, который необходим для работы сервиса. Подобные возможности есть только у оркестраторов-провайдеров, таких как Kubernetes. Слабой стороной Michman является то, что нельзя запросить любую топологию, необходимую пользователю. Эту проблему можно решить, если представить уровень инфраструктуры в Michman через TOSCA. Такое представление уже реализовано в оркестраторе уровня инфраструктуры - Clouni.

Еще из данных табл. 1 можно понять, что гибкое и продвинутое управление ЖЦ ПО является основным преимуществом систем оркестрации. Эти функции также необходимо реализовать в эффективном оркестраторе.

Табл. 1. Таблица сравнения систем Оркестрации
Tab. 1. Competitive analysis of orchestration systems

Инструмент	**Совместимость с провайдером**	**Возможность управления уровнем предоставления ресурсов**	**Уровень предоставления ресурсов**	**Имеется встроенный редактор топологий**	**Совместимость с инструментами конфигурации**
Инструменты, позволяющие осуществлять оркестрацию					
Terraform	OpenStack, AWS, Azure, GCP, Kubernetes, etc	IaaS, PaaS	-	Нет	-
Ansible	OpenStack, AWS, Azure, GCP, Kubernetes, etc	IaaS, PaaS, SaaS	-	Нет	-
Оркестраторы с поддержкой одного облачного провайдера					
Cloud Formation	AWS	IaaS, PaaS, SaaS, FaaS	IaaS, PaaS, SaaS, FaaS	да	
Heat Orchestrator	OpenStack, AWS	IaaS	-	Нет	
Michman	OpenStack	PaaS	PaaS	Нет	Ansible
Оркестраторы нескольких облачных провайдеров					
xOpera	OpenStack, AWS, Azure, GCP, Kubernetes	IaaS, SaaS, FaaS	-	Нет	Ansible
Cloudify	OpenStack, AWS, Azure, GCP, Kubernetes, etc	IaaS, PaaS	-	Да	Ansible, Terraform, etc
Alien4Cloud + Yorc	OpenStack, AWS, GCP, Kubernetes, etc	IaaS	-	Да	Terraform
Clouni	OpenStack, AWS, Kubernetes	IaaS	-	Нет	Ansible

Оркестраторы контейнеров					
Swarm	Есть возможность поставки при помощи других провайдеров	PaaS	-	Нет	Ansible, Terraform, etc
Kubernetes	Выступает как провайдер + есть возможность поставки при помощи других провайдеров	IaaS, PaaS	IaaS, PaaS	Нет	Ansible, Terraform, etc

Табл. 1. Продолжение 1
Table 1. Continuation 1

Инструмент	**Автоматический мониторинг**	**Валидация зависимостей**	**Использует ли TOSCA?**	**Автоматическое отслеживание состояния единиц развертывания**	**Возможность автоматического создания резервной копии и восстановления из нее**
Инструменты, позволяющие осуществлять оркестрацию					
Terraform	Нет	Нет	Нет	Нет	Да
Ansible	Нет	Нет	Нет	Нет	Да (не авт.-е)
Оркестраторы с поддержкой одного облачного провайдера					
Cloud Formation	Да	Нет	Нет	Да	Да
Heat Orchestrator	Нет	Нет	Да	Нет	Нет
Michman	Нет	Да	Планируется переход	Да	Нет
Оркестраторы нескольких облачных провайдеров					
xOpera	Нет	Нет	Да	Нет	
Cloudify	Да	Нет	Да	Да	
Alien4Cloud + Yorc	Да	Нет	Да	Да	
Clouni	Нет	Да	Да	Нет	
Оркестраторы контейнеров					
Swarm	Нет	Нет	Нет	Да	Нет
Kubernetes	Да	Нет	Нет	Да	Да

Табл. 1. Продолжение 2
Table 1. Continuation 2

Инструмент	Возможность автоматического возврата ПО к предыдущей версии	Возможность автоматического масштабирования	Возможность развертывание изменений для части сервисов
Инструменты, позволяющие осуществлять оркестрацию			
Terraform	Нет	Нет	Нет
Ansible	Да (не авт.-е)	Да (не авт.-е)	Да (не авт.-е)
Оркестраторы с поддержкой одного облачного провайдера			
Cloud Formation	Да	Да	Да
Heat Orchestrator	Нет	Нет	Нет
Michman	Нет	Да	Нет
Оркестраторы нескольких облачных провайдеров			
xOpera	Нет	Нет	Нет
Cloudify	Да	Да	Да
Alien4Cloud + Yorc	Нет	Нет	Нет
Clouni	Нет	Нет	Нет
Оркестраторы контейнеров			
Swarm	Да	Да	Нет
Kubernetes	Да	Да	Да

3.2 Проект решения

Далее будет описано решение по созданию единого оркестратора и будет описан требуемый функционал, но реализация не будет осуществлена в рамках работы.

Сравнительный анализ позволил выявить сильные и слабые стороны инструментов и функции, которые необходимо реализовать в оркестраторах, разработанных в ИСП РАН.

К тому же становится понятно, что не существует полнофункционального TOSCA оркестратора, в котором были бы полностью реализованы функции автоматизации.

Michman предоставляет сервисы PaaS по запросу как услугу. Однако для развертывания сервиса Michman использует заранее подготовленные скрипты развертывания инфраструктуры, которую нельзя дополнить и гибко настроить. Michman может развернуть один мастер узел и неограниченное количество узлов с рабочей нагрузкой в кластере. Во время удаления сервиса, Michman удаляет кластер целиком по причине того, что развертывание инфраструктуры и сервисов сильно связаны в инструменте через конфигурационные переменные. Этап создания инфраструктуры и этап развертывания

сервисов необходимо разделить. Тогда можно будет масштабировать инфраструктуру и сервисы не зависимо. Также необходимо реализовать поэтапное и раздельное развертывание и удаление.

Clouni позволяет унифицировано описывать единожды TOSCA template и развертывать инфраструктуру для нескольких облачных провайдеров. Однако в Clouni не реализована возможность предоставлять сервисы по запросу.

Выгодное решение – соединить оба оркестратора для совместной и эффективной работы.

Проектируемый TOSCA оркестратор должен принимать на вход унифицированный шаблон описания топологии инфраструктуры и сервисов. На выходе пользователь должен получить уже развернутую инфраструктуру или сервис в выбранном облачном провайдере с возможностью дальнейшей оркестрации и отслеживания статуса.

Функции оркестратора:

- графическое моделирование топологии;
- создание, удаление, обновление топологии;
- создание, удаление, обновление инфраструктуры для заданной топологии;
- отслеживание состояния единицы развертывания для заданной топологии;
- добавление нового провайдера;
- добавление добавление поддержки нового сервиса;
- валидация топологии и зависимостей в ней;
- автоматическое создание резервной копии единицы развертывания и восстановление из копии;
- автоматический возврат единицы развертывания к предыдущей версии;
- автоматическое масштабирование единицы развертывания;
- развертывание изменений на часть единиц развертывания.

Новый оркестратор будет ориентирован прежде всего на OpenStack и Ansible, однако в процессе проектирования необходимо закладывать возможность добавления поддерживаемых провайдеров и инструментов конфигурации.

В конечной реализации должны использоваться OpenStak-специфичные типы. Состояние сервисов должно быть реализовано в соответствии с дополнением к стандарту TOSCA Instance model [21]. Механизм подстановки параметров, необходимых для получения сценария для заданного инструмента конфигурации должен быть реализован через TOSCA substitute. После реализации этих требований можно будет реализовать механизм мониторинга

4 Заключение

В ходе описанной работы были решены следующие задачи:

- исследованы отличия оркестрации виртуальных машин с сервисами от оркестрации контейнеров;
- исследованы инструменты оркестрации контейнеров;
- исследованы инструменты конфигурации;
- исследованы механизмы оркестрации крупных облачных провайдеров;
- исследованы оркестраторы на базе унифицированного стандарта TOSCA;
- исследованы оркестраторы, разработанные в ИСП РАН;
- проведен сравнительный анализ методов построения облачных платформенных сервисов;
- предложен проект решения по объединению работы оркестраторов ИСП РАН.

Исходя из результатов работы были выявлены функциональные требования, необходимые для реализации оркестратора. В работе описан проект решения по объединению работы оркестраторов, разрабатываемых в ИСП РАН. Проект одобрен к реализации.

Реализация оркестратора, расширяемого путем добавления поддержки новых провайдеров и инструментов конфигурации, возможна при помощи стандарта TOSCA. Также, были проанализированы метрики эффективности внедрения инструментов оркестрации, которые можно будет использовать в дальнейшем.

Список литературы / References

[1] NIST Special Publication 500-332. The NIST Cloud Federation Reference Architecture. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-332.pdf.
[2] Topology and Orchestration Specification for Cloud Applications Version 1.0. OASIS Standard, 2013. Available at: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html.
[3] Bilgin I. Multi-Runtime Microservices Architecture. 2022. Available at: https://www.infoq.com/articles/multi-runtime-microservice-architecture/.
[4] Terraform About the Docs. 2022. Available at: https://www.terraform.io/docs.
[5] Red Hat Ansible Automation Platform. 2022. Available at: https://www.ansible.com/.
[6] Chef Documentation. 2022. Available at: https://docs.chef.io/.
[7] Puppet. 2022. Available at: https://puppet.com/.
[8] Ansible Galaxy. 2022. Available at: https://galaxy.ansible.com/.
[9] AWS CloudFormation. 2022. Available at: https://aws.amazon.com/ru/cloudformation/.
[10] Heat documentation. 2022. Available at: https://docs.openstack.org/heat/latest/.
[11] Michman. 2022. Available at: https://github.com/ispras/michman.
[12] Openstack. 2022. Available at: https://www.openstack.org/.
[13] Cloudify. 2022. Available at: https://github.com/cloudify-cosmo.
[14] X-opera. 2022. Available at: https://github.com/xlab-si/xopera-opera.
[15] Clouni TOSCA orchestrator. 2022. Available at: https://github.com/ispras/clouni.
[16] Alien4Cloud. 2022. Available at: http://alien4cloud.github.io.
[17] Yorc. 2022. Available at: https://github.com/ystia/yorc.
[18] Kubernetes Documentation. 2022. Available at: https://kubernetes.io/docs/home/.
[19] de Carvalho L.R., de Araujo A.P.F. Performance Comparison of Terraform and Cloudify as Multicloud Orchestrators. In Proc. of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), 2020, pp. 380-389.
[20] Qadeer A., Malik A. W. et al. Virtual Infrastructure Orchestration for Cloud Service Deployment, The Computer Journal, vol. 63, issue 1, 2020, pp. 295-307.
[21] Instance Model for TOSCA Version 1.0. Available at: http://docs.oasis-open.org/tosca/TOSCA-Instance-Model/v1.0/TOSCA-Instance-Model-v1.0.html. 2013.

Информация об авторах / Information about authors

Александра Андреевна БОРИСОВА – студентка магистратуры НИУ ВШЭ и старший лаборант отдела информационных систем ИСП РАН. Сфера научных интересов: облачные технологии и разработка ПО.

Alexandra Andreevna BORISOVA – student at the Higher School of Economics, research assistant of the Department of Information System of the Institute for System Programming of the RAS since 2020. Research interests: cloud technologies and software development.

Олег Дмитриевич БОРИСЕНКО – научный сотрудник отдела информационных систем и руководитель команды облачных технологий. Сфера научных интересов: облачные технологии и разработка ПО.

Oleg Dmitrievich BORISENKO is a specialist and team leader of the Department of Information System. Research interests: cloud technologies and software development.

DOI: 10.15514/ISPRAS-2022-34(5)-10

Data Mining Methods to Compare Englishes

O.V. Donina, ORCID: 0000-0002-1053-540X <olga-donina@mail.ru>
Voronezh State University,
1, Universitetskaya square, Voronezh, 394018, Russia

Abstract. The paper presents the results of the corpus-based research of noun cryptotypes in 20 varieties of English (Englishes). The data for this research collected from Mark Davies' corpora GloWbE and NOW enabled us to focus on variation in the covert classification of nouns in modern Englishes. A noun cryptotype introduced by Whorf is approached as 'a covert type of classification of nouns, marked by lexical selection in a syntactical classifier rather than a morphological tag'. The purpose of the study has been to compare and contrast the covert classification of basic 23 emotions in 20 Englishes (64,702 tokens). 20 Englishes have been clustered with the help of Data Mining methods (such as k-means clustering and a self-organizing Kohonen map). There are six clusters that appeared to be corresponding to geographic areas: American cluster (American and Canadian Englishes); Australian cluster (Australian and New Zealand Englishes); European cluster (British and Irish Englishes); Asian cluster (Indian, Pakistani, Singapore, Hong Kong, Malaysian, Bangladeshi, Sri Lankan, and Philippine Englishes); African cluster (Kenyan, South African, Nigerian, Ghanaian, and Tanzanian Englishes); Caribbean cluster (Jamaican English). The correlation coefficients among Englishes in the Asian and African clusters (the Outer Circle in the World Englishes Paradigm of Braj B. Kachru) range from 0.74 to 0.8 due to little contact among the varieties inside these clusters. The correlation coefficients between Englishes in the American, Australian and European clusters (the Inner Circle, Kachru) range from 0.92 to 0.933, which indicates a high consistency of these varieties owing to the long lasting, enduring linguistic contacts.

Keywords: Data Mining; computer modeling; corpora studies; cryptotype analysis; Englishes

For citation: Donina O.V. Data Mining Methods to Compare Englishes. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 163-170. DOI: 10.15514/ISPRAS-2022-34(5)-10

Методы интеллектуального анализа данных для сравнения диалектов английского языка

О.В. Донина, ORCID: 0000-0002-1053-540X <olga-donina@mail.ru>
Воронежский государственный университет,
Россия, 394018, Воронеж, Университетская площадь, 1

Аннотация. В статье представлены результаты корпусного исследования криптотипов имен существительных в 20 диалектах английского языка (Englishes). Данные для этого исследования, собранные из корпусов GloWbE и NOW Марка Дэвиса, позволили нам сосредоточиться на вариациях скрытой классификации существительных в современных диалектах английского языка. Криптотип существительного, введенный Уорфом, рассматривается как «скрытый тип классификации существительных, отмеченный лексическим отбором в синтаксическом классификаторе, а не морфологическим тегом». Цель исследования состояла в том, чтобы сравнить и сопоставить скрытую классификацию 23 основных эмоций в двадцати диалектах английского языка (64 702 токена). 20 диалектов английского языков были сгруппированы с помощью методов интеллектуального анализа данных (таких как кластеризация k-средних и самоорганизующаяся карта Кохонена). Шесть кластеров оказались соответствующими географическим областям: американский кластер (американский и канадский английский); австралийский кластер (австралийский и новозеландский диплекты английского языка); европейский кластер (британский и ирландский английский); азиатский кластер (индийский, пакистанский, сингапурский, гонконгский, малазийский, бангладешский, шри-ланкийский

и филиппинский диалекты английского); африканский кластер (кенийский, южноафриканский, нигерийский, ганский и танзанийский диалекты английского); карибский кластер (ямайский английский). Коэффициенты корреляции среди диалектов английского в азиатском и африканском кластерах (внешний круг в парадигме Браджа Б. Качру) колеблются от 0,74 до 0,8 из-за небольшого контакта между диалектами внутри этих кластеров. Коэффициенты корреляции между диалектами в американском, австралийском и европейском кластерах (внутренний круг) колеблются от 0,92 до 0,933, что свидетельствует о высокой согласованности этих диалектов за счет длительных, устойчивых языковых контактов.

Для цитирования: Донина О.В. Методы интеллектуального анализа данных для сравнения диалектов английского языка. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 163-170. DOI: 10.15514/ISPRAS-2022-34(5)-10

1. Introduction

Due to the universal digitalization observed in the modern information society, a lot of new scientific fields are emerging, e.g. Digital Humanities (DH) or eHumanities, which is an innovative interdisciplinary field of research that combines methods of the humanities, social and computer sciences intending to explore the possibilities of applying new digital technologies in the humanities. Qualitative data analysis in these sciences can be improved mainly through the digitized texts available for research. It is worth noting the availability and manufacturability of full-text archives (for example, various national corpora), which, instead of a small manual sample (n <100), allow to analyze a statistically representative subset (n> 1,000) or a whole corpus (n> 100,000). The computational linguistics approach allows the researcher to work with large amounts of data and at the same time pay more attention to linguistic details by modeling and visualizing the results obtained.

In sciences, where the object of research is inaccessible to direct observation, such as linguistics, it becomes necessary to model it using various means of visualizing the object under study. Moreover, due to the complexity of studying such a dynamic and multifaceted phenomenon as language, it is advisable, as research in recent years has shown, to use cognitive modeling tools. A review of modern theories of cognitive modeling in linguistics was made by L.S. Abrosimova [1]. The review discusses the success of this approach, including the applications in computational linguistics (for example, in the creation of artificial machine languages and in the improvement of automated translation). In our current work, we use computer cognitive modeling of unobservable objects, using a set of Data Mining methods, the essence of which is the process of discovering new interpretations of knowledge in raw data that are necessary for making decisions in various spheres of human activity with the help of the methods of mathematical statistics. Data Mining is a multidisciplinary field that has arisen and is developing based on such sciences as applied statistics, pattern recognition, artificial intelligence, and database theory.

2. Methodology and related work

Our research methodology of linguistic categories in different linguistic environments is based on three components: cryptotype analysis, methods of corpus linguistics, and Data Mining. We have analyzed noun cryptotypes, i.e. the language categories hidden in the English language. These hidden classes of nouns were described in the works of O.O. Boriskina [2, 3, 4, 5, 6, 7].

A noun cryptotype introduced by Whorf [15] is approached as ‘a covert type of classification of nouns, marked by lexical selection in a syntactical classifier rather than a morphological tag’ [16]. The purpose of the study has been to compare and contrast the covert classification of basic 23 emotions in 20 Englishes (64,702 tokens).

Noun Cryptotypes represent word classes organized on two main principles: the modeling power of the Cryptotype Core nouns and the inherent potentialities of the Cryptotype Periphery nouns to imitate the Core nouns syntactic behavior, i.e. to borrow the cryptotype Core nouns classifiers and thus to adapt to the Core nouns combinatory characteristics. To illustrate, in context An authentic

feeling was able to penetrate the structure of the debate, the S-position of the verb to penetrate can be substituted by a name of a sharp-pointed object. The noun feeling substitutes the above-mentioned position. This metaphor is meant as the discourse evidence of the noun question belonging to the periphery of cryptotype 'Sharp objects'.

At the moment, 6 cryptotypes of the English language have been identified and described (they correspond with explicit lexical and grammatical categories of some other world languages according to a typological research): the cryptotype Res Liquidae (a class of liquids, the prototype is 'water'), Res Acutae (a class of sharp objects, the prototype is 'thorn'), Res Filiformes (a class of thin objects of unstable form, the prototype is 'thread'), Res Rotundae (a class of round objects, the prototype is 'ball'), Res Parvae (a class of hand-fitting objects, the prototype is 'apple'), Res Longae Penetrantes (a class of solid, long, pointed objects, the prototype is 'stick') [8, 11, 12]. For example, the nominal cryptotype of the English language Res Liquidae includes such nouns as water, blood, milk, and other nominations of objects of reality that exist in a liquid state. These nouns are the prototypes of the cryptotype. At the same time, this cryptotype also contains nouns denoting abstract concepts. Concepts such as life, goodness, or passion do not occur in a liquid state, but a person often categorizes them by analogy with a liquid one. Thus, the names of abstract semantics in metaphorical use can be included in the nominative cryptotypes distinguished in the English language, so that the names of concrete and abstract semantics coexist in the same language class.

Cryptotype analysis was carried out on the names of emotional state and sensory experience (such as anger, fear, love, etc.) in 20 varieties of the English language (Englishes) presented in the Mark Davis' corpus [9]. Along with British and American, it also features rare varieties (for example, Kenyan or Tanzanian Englishes). The volume of the research corpus, formed based on the results of semi-automatic processing of corpus queries, amounted to 65,000 tokens. Previously, to visualize the results of a cryptotype study, Chernov's faces were used [10]. But in our research, given the rather large volume of the resulting research corpus and the need to bring together parameters of different quality (namely, 20 variants of the English language under consideration, 23 names of emotions, and 6 nominal cryptotypes), the use of this method turned out to be impossible. That is why we decided to try to apply the methods of Data Mining and computer-cognitive modeling to achieve our goal.

3. The used approach

First of all, to determine the significance of the factors and the possible reduction of the input parameters before clustering using the Deductor Academic 5.3 program, a factor analysis based on the "varimax" method was carried out. Six currently allocated cryptotypes were used as input data.

The next step was the k-means clustering, as well as the construction of a self-organizing Kohonen map, which is a type of neural network algorithms. Clustering is used to distribute a set of objects into classes that are not initially specified, with the k-means algorithm being the most commonly used. Artificial neural networks, which arose as a model of the biological nervous system, consist of input, hidden and output layers of neurons, and for the input and output layers the parameters are known, while implicit signal transformations take place in the hidden one. In linguistics, neural networks are used in neural network models of a language, in machine translation, automatic clustering of vocabulary (Kohonen maps), etc.

Having employed factor analysis, it was shown that as a result of the performed rotation, there were no significant changes in the structure of the factor space (i.e., the factors set automatically correspond to six cryptotypes by 96.09% - 99.51% (Table 1)). Thus, we can talk about the stability of the data, which indicates the independence of the factors reflected in the correlation matrix, i.e. the high explanatory significance of all factors (cryptotypes) and the possibility of using them for further clustering were proved.

Table 1. Results of factor analysis

	Final factors (Varimax method)					
	Factor 1	**Factor 2**	**Factor 3**	**Factor 4**	**Factor 5**	**Factor 6**
Res Acutae			0,9951			
Res Filiformes		0,9841				
Res Liquidae						0,9609
Res Longae Penetrantes					0,9769	
Res Parvae				0,9851		
Res Rotundae	0,9849					

Using the Kohonen neural network, the input maps of the neurons of six cryptotypes were generated, i.e. the internal structure of the input data was visualized by adjusting the weights of the neurons of the maps, where the areas containing approximately the same inputs for the analyzed examples are marked with a certain color. As a result of vector quantization, individual varieties of the English language were grouped into six clusters corresponding to geographic areas (Fig. 1): 1. American area (American English and Canadian English), 2. Australian area (Australian English and New Zealand English), 3. European area (British English and Irish English), 4. Asian area (Indian English, Pakistani English, Singapore English, Hong Kong English, Malaysian English, Bangladeshi English, Sri Lankan English, and Philippine English), 5. African area (Kenyan English, South African English, Nigerian English, Ghanaian English, and Tanzanian English), 6. Caribbean area (Jamaican English).

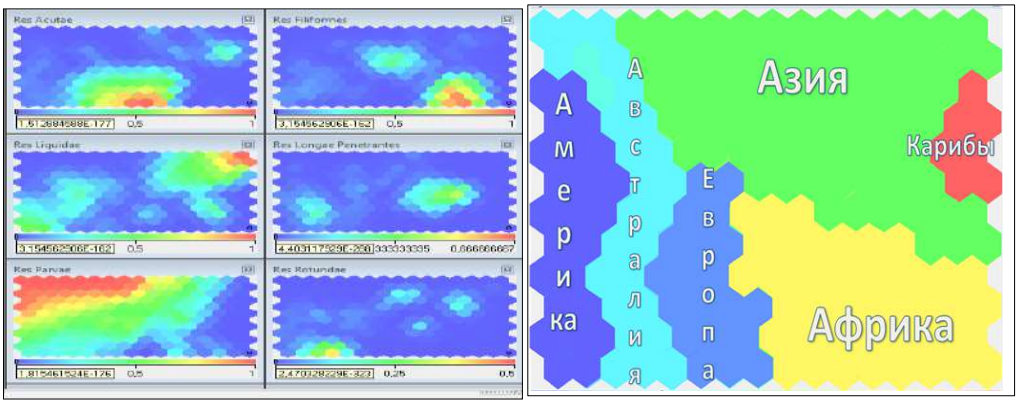


Fig. 1. Kohonen self-organizing maps

The coincidence of the cryptotype categorization with the geographical one emphasizes the importance of the areal influence noted in the work of V.N. Polyakova and E.I. Yaroslavtseva [13]. Within the framework of this work (on the material of the database "Languages of the World"), the phenomenon of a typological shift is studied, the essence of which is that languages in the process of areal contacts acquire new typological features and lose some of the existing ones. At the same time, widespread signs tend to further spread, and low-frequency ones tend to wash out [13: 114-115].

According to the cryptotype analysis for all 23 names of emotions in 20 Englishes, statistics on cryptotypes were calculated, which reflected the general trends characteristic of the research corpus as a whole (Fig. 2). The data obtained showed that in 92.3% of cases the share of the cryptotype Res Rotundae does not exceed 5.6%, which is the lowest value among the cryptotypes, i.e. this class is the least represented in our research corpus. In 93.6% of examples, the representation of the cryptotype Res Longae Penetrantes is in the range from 0% to 11.1%, i.e. it appears next to last in terms of representation. The fourth most common cryptotype is Res Filiformes: in 96.1% of cases, the share of this cryptotype varies from 0% to 22.2%. The cryptotype Res Acutae is the third in frequency: in 79.3% of cases its share of representation does not rise above 22.2%, but at the same time in 3% of examples it is one of the most represented with a value of the cryptotype activity of some emotion names from 88, 9% to 100%. The second place is taken by the cryptotype Res Liquidae, whose share of representation in 94.2% of cases ranges from 0% to 55.6%. The leader in the prevailing majority of word usage is the cryptotype Res Parvae, which share of representation in 55.2% of cases exceeds 55.6%.

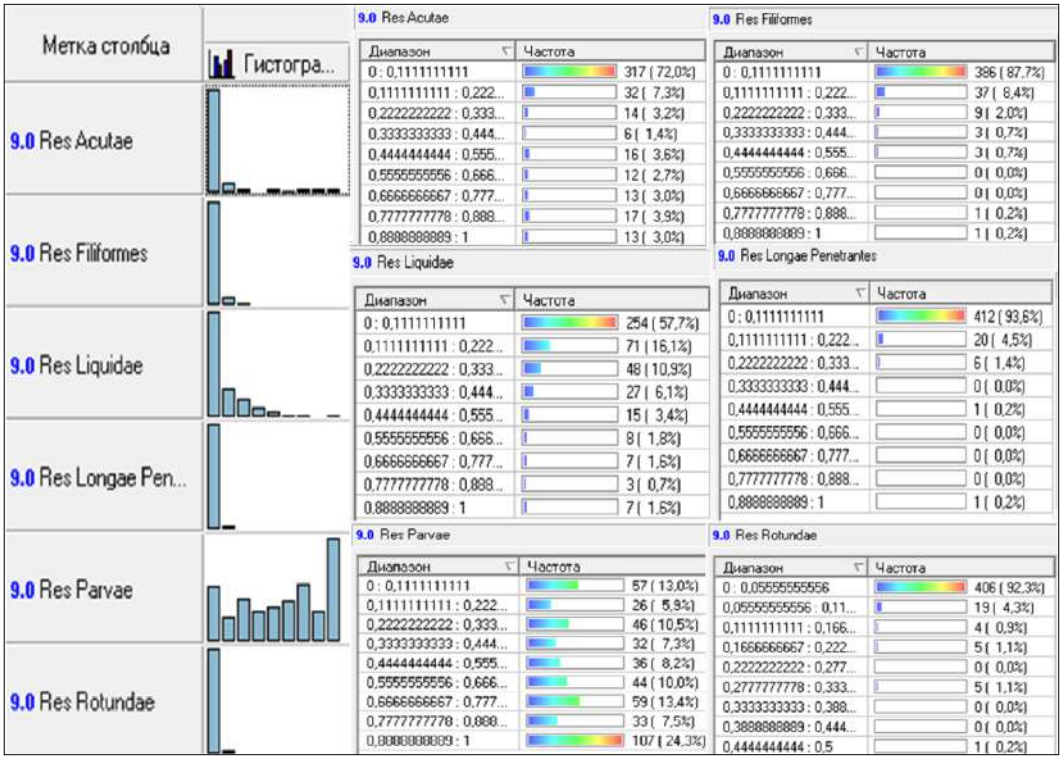


Fig. 2. Frequency of cryptotype representation in the research building

The next step in our research was to create rules based on a decision tree that would allow us to write a computer program capable of establishing the areal affiliation of a language dialect. Decision trees are a method of automatic data analysis that forms a sequential structure of rules, where each object corresponds to a single node that gives a solution. The results of such an analysis can be presented both in the form of a hierarchy (Fig. 3) and in the form of a set of rules describing classes. In the future, thanks to the compiled rules and descriptions of clusters, it becomes possible to trace the dynamics of the influence of language varieties/language areas on each other, conducting a similar study in 10-15 years.

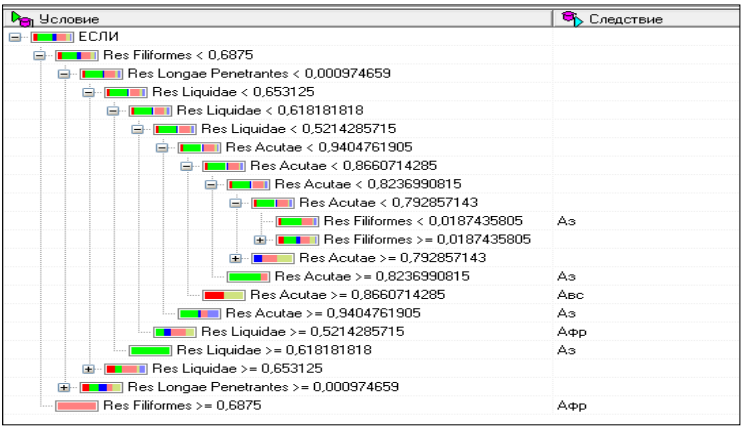


Fig. 3. Fragment of a decision tree

2. Results and conclusion

Data Mining Methods helped to obtain the following main results:

1) All emotions are attributed to one of the six noun cryptotypes in all Englishes:
   - Res Parvae – a class of hand-fitting objects (prototype: stone),
   - Res Liquidae – a class of liquids (water),
   - Res Filiformes – a class of thin objects of unstable form (thread),
   - Res Rotundae – a class of round objects (ball),
   - Res Longae Penetrantes – a class of solid, long, pointed objects (spear),
   - Res Acutae – a class of sharp objects (thorn).

1) 20 Englishes have been clustered with the help of Data Mining methods (such as k-means clustering and a self-organizing Kohonen map). There are six clusters that appeared to be corresponding to geographic areas: American cluster (American and Canadian Englishes); Australian cluster (Australian and New Zealand Englishes); European cluster (British and Irish Englishes); Asian cluster (Indian, Pakistani, Singapore, Hong Kong, Malaysian, Bangladeshi, Sri Lankan, and Philippine Englishes); African cluster (Kenyan, South African, Nigerian, Ghanaian, and Tanzanian Englishes); Caribbean cluster (Jamaican English).
2) The correlation coefficients among Englishes in the Asian and African clusters (the Outer Circle in the World Englishes Paradigm of Braj B. Kachru [14]) range from 0.74 to 0.8 due to little contact among the varieties inside these clusters.
3) The correlation coefficients between Englishes in the American, Australian and European clusters (the Inner Circle, Kachru [14]) range from 0.92 to 0.933, which indicates a high consistency of these varieties owing to the long lasting, enduring linguistic contacts.
4) The most relevant to the metaphorical categorization of emotions in all Englishes is Res Parvae. Notably, there is a grammatical category of hand-fitting objects in the grammar systems of some indigenous languages of African, American and Australian clusters, and there is a classifier for counting hand-fitting objects in some Asian languages.
5) Res Liquidae is the second cryptotype frequently associated with emotions for English-speakers of Australian, American, African and Caribbean clusters, whereas for Asian and European clusters it is the class of sharp objects. Presumably, it could be due to close contacts of the English language with the languages of the indigenous population.

References / Список литературы

[1] Abrosimova L.S. Word formation in the linguistic categorization of the world. Rostov-on-Don, SFU Publishing House, 2015, 328 p. (in Russian) / Абросимова Л.С. Словообразование в языковой категоризации мира. Ростов на Дону, Изд-во ЮФУ, 2015 г., 328 стр.

[2] Boriskina O.O Linguistic categorization of the elements. In Proc. of the 2nd International Conference on Philology and Culture, 2019, pp. 149-157 (in Russian) / Борискина О.О. Языковая категоризация стихий. Материалы 2-й международной конференции «Филология и культура», 1999 г., стр. 149-156.

[3] Boriskina O.O. Cryptoclasses of primary elements as an element of the ontognostic description of language. In Problems of linguistic prognostics, issue 1, Voronezh, Central Black Earth Book Publishing House, 2000, pp. 121-126 (in Russian) / Борискина О.О. Криптоклассы первостихий как элемент онтогностического описания языка. В сборнике статей «Проблемы Лингвистической Прогностики», вып. 1, Воронеж, Центрально-Черноземное книжное издательство, 2000 г., стр. 121-126.

[4] Boriskina O.O. National-specific linguistic consciousness and borrowed word. In Intercultural communication and problems of national identity, Voronezh, Voronezh State University Press, 2002, pp. 406-410 (in Russian) / Борискина О.О. Национально-специфическое языковое сознание и заимствованное слово. В сборнике статей «Межкультурная коммуникация и проблемы национальной идентичности», Воронеж, издательство Воронежского государственного университета, 2002 г., стр. 406-410.

[5] Boriskina O.O. The dynamic noun combinatory profile. Issues of cognitive linguistics, 2008, issue 3, pp. 64-69 (in Russian) / Борискина О.О. Моделирование синтагматической динамики слова. Вопросы когнитивной лингвистики, вып. 3, 2008 г., стр. 64-69. .

[6] Boriskina O.O. Cryptotype projection of abstract entities: applications of cryptotype approach to noun combinations study. Proceedings of Voronezh State University. Series: Linguistics and intercultural communication, issue 1, 2009, pp. 32-37 (in Russian) / Boriskina O.O. Криптоклассные проекции мира непредметных сущностей: опыт криптоклассного анализа словосочетаемости. Вестник ВГУ. Серия: Лингвистика и межкультурная коммуникация, вып. 1, 2009 г., стр. 32-37.

[7] Boriskina O.O. Explanation of the unexplained or about the motivation of the unmotivated. Vestnik of Saint Petersburg University. Series 9. Philology. Oriental studies. Journalism, issue 1, 2010, pp. 95-100 (in Russian) / Борискина О.О. Объяснение необъяснимого или мотивация немотивированного. Вестник Санкт-Петербургского университета. Серия 9. Филология. Востоковедение. Журналистика, вып. 1, 2010 г., стр. 95-100.

[8] Boriskina O.O., Marchenko T. An Algorithm for Analysis of Distribution of Abstract Nouns in Cryptotypes. In Proc. of the 2010 International Conference on Artificial Intelligence, 2010, pp. 907-913.

[9] NOW Corpus (News on the Web). Available at: https://www.english-corpora.org/now/.

[10] Donina O.V. The study of cryptopytes: vizualization of reserach results. Proceedings of Voronezh State University. Series: Linguistics and intercultural communication, issue 3, 2015, pp. 105-112 (in Russian) / Донина О.В. Способы визуализации результатов криптоклассного исследования. Вестник ВГУ. Серия: Лингвистика и межкультурная коммуникация, вып. 3, 2015 г., стр. 105-112.

[11] Donina O.V., Boriskina O.O. Emotive lexemes from the perspective of areal variativity. Proceedings of Voronezh State University. Series: Linguistics and intercultural communication, issue 4, 2016, pp. 41-45 (in Russian) / Донина О.В., Борискина О.О. Эмотивная лексика в аспекте ареальной вариативности. Вестник ВГУ. Серия: Лингвистика и межкультурная коммуникация, вып. 4, 2016 г., стр. 41-45.

[12] Kretov A.A., Boriskina O.O, Vasilyeva N. 2004. «Flight of thought» and methods of cryptoclass investigation. Proceedings of Voronezh State University. Series: Linguistics and intercultural communication, issue. 1, 2004, pp. 61-65 (in Russian) / Кретов А.А., Борискина О.О., Васильева Н.Е. «Полёт мысли» и методика исследования криптоклассов. Вестник ВГУ. Серия: Лингвистика и межкультурная коммуникация, вып. 1, 2004 г., стр. 61-65.

[13] Polyakov V.N., Yaroslavtseva E.I. The Quantitative Parameters of Typological Shift. Scientific notes of Kazan State University. Series: Humanitarian sciences, vol. 150, issue 2, 2008, pp. 97-118 (in Russian) / В.Н. Поляков, Е.И. Ярославцева. Квантитативные закономерности типологического сдвига в языках Евразии (на материале БД «Языки мира» ИЯ РАН). Ученые записки Казанского университета. Серия Гуманитарные науки, том 150, вып. 2, 2008 г., стр. 97-118.

[14] Kachru B. Models for Non–native Englishes. The Other Tongue: English across cultures. Urbana: University of Illinois Press, 1992, 416 p.

[15] Whorf B.L. Language, Thought and Reality. Selected Writings of Benjamin Lee. The MIT Press, 1964, 290 p.
[16] Boriskina O.O. The Main Criteria for the Exploration of Noun Cryptotypes. In Proc. of the VI International Scientific Conference on Language, Culture, Society, 2011. Available at: http://www.mosinyaz.com/conferences/mnk6_s3_12/.

Information about the author / Информация об авторе

Olga Valer'evna DONINA – Candidate of Philological Sciences, Associate Professor of the Department of Theoretical and Applied Linguistics. Research interests: computational linguistics, corpus linguistics, cognitive linguistics, metaphor study, the study of Englishes.

Ольга Валерьевна ДОНИНА – кандидат филологических наук, доцент кафедры теоретического и прикладного языкознания. Научные интересы: компьютерная лингвистика, корпусная лингвистика, когнитивная лингвистика, метафороведение, изучение диалектов английского языка.

DOI: 10.15514/ISPRAS-2022-34(5)-11

Контекстное разрешение омонимии на основе центроидно-контекстной модели

[1,2,3] *Александр А. Хорошилов, ORCID: 0000-0003-4885-3232 <khoroshilov@mail.ru>*
[1,4] *Ю.В. Никитин, ORCID: 0000-0002-7641-0247 <yuri.v.nikitin@gmail.com>*
[2,5] *А.В. Кан, ORCID: 0000–0001–9410–406X <kanav@nrczh.ru>*
[4] *Я.Д. Козловская, ORCID: 0000–0002–1780–5687 <yana_kozlovskaia@mail.ru>*
[2] *Е.А. Евдокимова, ORCID: 0000-0003-4719-2786 <evdokimovaekan@mail.ru>*

[1] *Федеральный исследовательский центр «Информатика и управление» РАН, 119333, Россия, г. Москва, Вавилова, д.44, кор.2*
[2] *Московский авиационный институт (национальный исследовательский университет), 125993, Россия, г. Москва, Волоколамское шоссе, д. 4*
[3] *27-й Центральный научно-исследовательский институт МО РФ 123007, г. Москва, 1-й Хорошевский пр-д, д. 5*
[4] *Научно-промышленная компания «Высокие технологии и стратегические системы», 107023, Россия, Москва, ул. Электрозаводская, д. 27, стр. 9*
[5] *Национальный исследовательский центр «Институт имени Н.Е. Жуковского», 125319, Россия, г. Москва, ул. Викторенко, д.7*

Аннотация. В статье описывается новый метод контекстного разрешения омонимии на основе центроидно-контекстной модели (ЦКМ). Предлагаемый метод выявления случаев омонимии в корпусе текстов и ее разрешения с помощью модели ЦКМ базируется на теоретической концепции фразеологического концептуального анализа текстов (ФКАТ) и уникальной машинной грамматике, в основу которой положена система флективных классов русских слов. Заложенное в теоретической концепции флективных классов слов русского языка жесткое соответствие между формой представления слов и их грамматической информацией позволило создать на этой основе новые классы – классы слов, имеющие одинаковые наборы грамматических признаков, соответствующие их формам представления в сходных контекстных окружениях. При разработке этой модели авторы исходили из следующей гипотезы: одинаковым последовательностям обобщенных символов классов слов (обобщенным синтагмам) должны соответствовать одинаковые синтаксические структуры различных фрагментов текстов. При этом предполагалось, что такая гипотеза верна для любых синтаксических моделей и может быть полезна при решении как глобальных, так и частных задач анализа текста. С помощью этого метода было предложено новое решение задачи разрешение омонимии на основе предлагаемой модели ЦКМ.

Ключевые слова: разрешение омонимии; омонимы; центроидно-контекстная модель; обобщенные синтагмы

Для цитирования: Хорошилов Александр А., Никитин Ю.В., Кан А.В., Козловская Я.Д., Евдокимова Е.А. Контекстное разрешение омонимии на основе центроидно-контекстной модели. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 171-182. DOI: 10.15514/ISPRAS-2022-34(5)-11

Context resolution of homonymy based on a centroid-context model

[1,2,3] *Alexander A. Khoroshilov, ORCID: 0000-0003-4885-3232 <khoroshilov@mail.ru>*
[1,4] *Yu.V. Nikitin, ORCID: 0000-0002-7641-0247 <yuri.v.nikitin@gmail.com>*
[2,5] *A.V. Kan, ORCID: 0000–0001–9410–406X <kanav@nrczh.ru>*
[4] *Ya.D. Kozlovskaya, ORCID: 0000–0002–1780–5687 <yana_kozlovskaia@mail.ru>*
[2] *E.A. Evdokimova, ORCID: 0000-0003-4719-2786 <evdokimovaekan@mail.ru>*

[1] *Federal Research Center «Computer Science and Control» of the Russian Academy of Sciences, 44 building 2, Vavilova st., Moscow, 119333, Russia*
[2] *Moscow Aviation Institute (National Research University), 4, Volokolamskoye Shosse, Moscow, 125993, Russia*
[3] *27th Central Research Institute of the Ministry of Defence of the Russian Federation 5 1st Khoroshevsky Passage, 123007, Moscow, Russia*
[4] *Scientific and Industrial Company «High Technologies and Strategic Systems», 27 building 9, Electrozavodskaya st., Moscow, 107023, Russia*
[5] *National Research Center «Zhukovsky Institute», 7, Viktorenko st., Moscow, 125319, Russia*

Abstract. The article describes a new method for contextual resolution of homonymy based on a centroid-context model (CCM). The proposed method of detecting cases of homonymy in the corpus of texts and its resolution using the CCM model is based on the theoretical concept of phraseological conceptual analysis of texts (FCAT) and unique machine grammar, which is based on a system of inflective classes of russian words. The rigid conformity between the form of presentation of words and their grammatical information laid down in the theoretical concept of inflective classes of words of the Russian language made it possible to create on this basis new classes - classes of words that have the same sets of grammatical features, conforming to their forms of representation in similar contextual environments. When developing this model, the authors proceeded from the following hypothesis: the same sequences of generalized characters of word classes (generalized syntagms) should correspond to the same syntactic structures of various fragments of texts. At the same time, it was assumed that such a hypothesis is true for any syntactic models and can be useful in solving both global and particular problems of text analysis. Using this method, a new solution to the problem of resolving homonymy based on the proposed CCM model was proposed.

Ключевые слова: homonymy resolution; homonyms; centroid-context model; generalized syntagms

For citation: Khoroshilov Alexander A., Nikitin Yu.V., Kan A.V., Kozlovskaya Ya.D., Evdokimova E.A. Context resolution of homonymy based on a centroid-context model. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 171-182 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-11

1. Введение

В процессе автоматической обработки текстов нередко возникает ситуация, когда одни и те же словоформы в различных контекстных окружениях принимают разные смысловые значения. Такое явление в лингвистике называется омонимией. Разрешение (снятие) омонимии является одной из важнейших проблем автоматической обработки естественного языка. Решение этой проблемы необходимо для многих приложений компьютерной лингвистики, в частности, для интеллектуальных систем обработки текстов при формализации смысловой структуры текстов и построении их формальной модели метаданных, для поисковых систем для повышения точности обработки некоторых классов запросов и ряда других аналогичных задач.

В настоящее время наметилось три основных подхода к решению проблемы снятия омонимии: а) подход, основанный на правилах, б) подход, основанный на статистике, в) подход, основанный на машинном обучении.

Суть метода, основанного на правилах, сводится к тому, что в некоторых ситуациях анализ контекста помогает понять синтаксическую структуру части предложения, а с ее помощью и

формы слов. Например, в конструкции вида: *ни ..., ни ...* оба слова обычно принадлежат одной и той же части речи и находятся в одной и той же форме. Если одно из слов окажется неомонимичным, определить форму второго несложно. Этот метод требует ручного составления правил, для каждого из правил требуется написать самостоятельный программный модуль.

Подсчёт статистики различных вариантов разбора по корпусу является простейшим способом снятия морфологической омонимии. При этом по размеченному корпусу со снятой омонимией происходит вычисление апостериорных вероятностей каждого из разборов. В современных системах анализа текстов на естественном языке применяются несколько способов подсчёта таких вероятностей, однако большее влияние на точность снятия омонимии оказывает сам корпус: его представительность, объём, точность разметки. Для разрешения омонимии может быть реализован метод простого подсчёта вероятности для каждого из набора тегов и слов корпуса. Более точные результаты даёт учёт контекста слова.

Для решения проблемы снятия омонимии используются такие методы машинного обучения, как скрытые марковские модели [1], условные случайные поля [2], рекуррентные нейронные сети [3] и др. (подробнее см. [4]). Для обучения метода классификации также используются размеченные корпуса, например, НКРЯ (национальный корпус русского языка) с вручную снятой омонимией. Для контекстного снятия омонимии может использоваться, например, метод CRF (Conditional Random Field, условные случайные поля). Этот метод хорош в задачах определения части речи, определения именованных сущностей и др. [5]. CRF является дискриминативной вероятностной моделью. Одним из главных достоинств этой модели является то, что она не требует моделировать вероятностные зависимости между так называемыми наблюдаемыми переменными.

В настоящей работе описан метод, позволяющий предиктивно, на основе принципа лингвистической аналогии выявлять словоформы, принадлежащие к двум или более словоизменительным парадигмам и назначать им соответствующие грамматические характеристики. В словарном комплексе эти словоформы помечаются как омонимы и обе словоформы с грамматической информацией помещаются в соответствующие словари. В процессе автоматического анализа текстов на основе применения центроидно-контекстных моделей (ЦКМ) разрешается конкретная языковая ситуация по ее контекстному окружению. Рассмотрим более подробно этапы выявления омонимичных словоформ и разрешения случаев омонимии на основе метода контекстного разрешения омонимии.

2. Выявление омонимичных словоформ русского языка

В соответствии с теоретической концепцией фразеологического концептуального анализа текстов (ФКАТ) [6-8] некоторые виды омонимии словоформ русского языка частично разрешаются с помощью тематических концептуальных словарей, в которых значения слов устанавливаются в соответствии с их доминантными значениями, принятыми в конкретной тематической области и зафиксированные в составе терминологических словосочетаний. Поэтому основное внимание необходимо сосредоточить на методах автоматического выявления и разрешения грамматической омонимии для отдельных словоформ.

В основу метода выявления грамматической омонимии словоформ была положена гипотеза, что *в достаточно большом корпусе политематических текстов должна отобразиться значительная часть словоформ, имеющих один и тот же буквенный состав, но принадлежащих к различным грамматическим словоизменительным парадигмам*. Другими словами, если для всех словоформ достаточно большого корпуса текстов автоматически построить их словоизменительные парадигмы и назначить им грамматические признаки исходной словоформы, то все словоформы, содержащие одинаковый буквенный состав, но имеющие различные грамматические характеристики с большой вероятностью будут являться ***грамматическими омонимами***.

В качестве базовых признаков, на основе которых возможно идентифицировать каждую текстовую словоформу и автоматически построить ее грамматическую парадигму были выбраны номер флективного класса словоформы и ее грамматическое окончание. Это сочетание признаков однозначно характеризует смысловое содержание словоформы и ее конкретную форму в контекстном окружении. При этом мы исходили из того, что флективный класс словоформы однозначно соотносится с грамматическим классом словоформы, типом ее словоизменения, а в связке с грамматическим окончанием однозначно устанавливается расширенный состав грамматических характеристик конкретной словоформы. Рассмотрим эту идею в виде модели процесса выявления омонимов в корпусе тестов.

Пусть имеются две упорядоченные последовательности словоформ $\{w_i^1\}_{i=1}^{12}$ и $\{w_i^2\}_{i=1}^{12}$, где w_1^1 и w_1^2 – первая и вторая исходные словоформы соответственно, а $w_2^1, w_3^1, \dots, w_{12}^1$ и $w_2^2, w_3^2, \dots, w_{12}^2$ – их словоизменительные парадигмы. Имеются две упорядоченные последовательности $\{p_i^1\}_{i=1}^{12}$ и $\{p_i^2\}_{i=1}^{12}$, содержащие в себе признаки словоформ из последовательностей $\{w_i^1\}_{i=1}^{12}$ и $\{w_i^2\}_{i=1}^{12}$ соответственно. Признак p_i включает в себя информацию о флективном классе, окончании и грамматической информации (род, число, падеж, лицо) $p_i^j = \{FK_i^j, OK_i^j, GI_i^j\}$. Последовательности $\{p_i^1\}_{i=1}^{12}$ и $\{p_i^2\}_{i=1}^{12}$ обладают свойством:

$$\exists i: p_i^1 \neq p_i^2, i \in \{2,3,\dots,12\}.$$

Проблема разрешения омонимии возникает, когда встречаются $w_i^1 = w_j^2$, $i, j \in \{1,2,\dots,12\}$.

Подтверждением выше приведенной гипотезы служит пример, приведенный в табл.1, в которой приведены словоизменительные парадигмы для слов: «суд» и «судно».

Табл. 1. Автоматически сформированные грамматические словоизменительные парадигмы текстовых словоформ
Table 1. Automatically generated grammatical inflectional paradigms of text word forms

Исходная словоформа:	***Исходная словоформа***:
суд ОК=00, FK=001	судно ОК=01, FK=071
Словоизменительная парадигма №1:	***Словоизменительная парадигма №2:***
Им.п., ед.ч. = суд ОК=00, FK=000	Им.п., ед.ч. = судно ОК=00, FK=071
Род.п., ед.ч. = **суда ОК=01, FK=001**	Род.п., ед.ч. = судна ОК=01, FK=071
Дат.п., ед.ч. = суду ОК=01, FK=001	Дат.п., ед.ч. = судну ОК=01, FK=071
Вин.п., ед.ч. = суд ОК=00, FK=001	Вин.п., ед.ч. = судно ОК=00, FK=071
Тв.п., ед.ч. = судом ОК=02, FK=001	Тв.п., ед.ч. = судном ОК=02, FK=071
Пр.п., ед.ч. = суде ОК=01, FK=001	Пр.п., ед.ч. = судне ОК=01, FK=071
Им.п., мн.ч. = суды ОК=01, FK=001	Им.п., мн.ч. = суда ОК=01, FK=071
Род.п., мн.ч.= **судов ОК=02, FK=001**	Род.п., мн.ч.= судов ОК=02, FK=071
Дат.п., мн.ч.= **судам ОК=02, FK=001**	Дат.п., мн.ч.= **судам ОК=02, FK=071**
Вин.п., мн.ч.= суды ОК=01, FK=001	Вин.п., мн.ч.= **суда ОК=01, FK=071**
Тв.п., мн.ч.= **судами ОК=03, FK=001**	Тв.п., мн.ч.= **судами ОК=03, FK=071**
Пр.п., мн.ч.= **судах ОК=02, FK=001**	Пр.п., мн.ч.= **судах ОК=02, FK=071**

Из табл. 1 видно, что словоформы «***суда***», «***судах***» «***судам***» «***судами***» принадлежат к двум различным грамматическим парадигмам (№1 и №2), в которых эти словоформы имеют разные грамматические признаки. Например, для словоформы «суда» в парадигме №1 эта словоформа имеет характеристики: FK=001, OK=01, в парадигме №2 та же словоформа имеет характеристики: FK=071, OK=01.

В качестве тестового корпуса текстов был выбран массив сообщений СМИ объемом 740 тыс. документов и содержащий более 12 млн. слов. Далее обработка этого корпуса текстов производилась по следующей технологической схеме:

Этап 1. По этому корпусу текстов был построен частотный словарь словоформ, включающий 880 тыс. разных словоформ.

Этап 2. Все словоформы были обработаны процедурой морфологического анализа, и им был назначен набор грамматических характеристик, из которых для дальнейшей обработки были выбраны только две характеристики – номер FK и OK.

Этап 3. Для каждой словоформы корпуса была построена ее грамматическая словоизменительная парадигма и всем членам парадигмы были назначены грамматические признаки (FK и OK) той словоформы, на основе которой была построена ее грамматическая словоизменительная парадигма (см. табл. 1).

Этап 4. Все автоматически сгенерированные члены парадигмы всех словоформ корпуса были слиты в один массив и отсортированы. Далее были исключены все словоформы с одинаковыми грамматическими характеристиками (FK и OK) и сохранены только словоформы, имеющие различные грамматические характеристики.

Этап 5. Полученный массив рассортирован по грамматическим характеристикам, исключены дублирующие записи с одинаковыми характеристиками и назначены каждому типу омонимии его трансформационную модель (табл. 2).

3. Создание словарей ЦКМ для разрешения омонимии

Для реализации процесса разрешения омонимии в текстовых документах необходимо создать декларативные средства для решения этой задачи. Как было выше указано разрешение омонимии словоформы возможно только с учетом ее контекстного окружения. Разработанная авторами центроидно-контекстная модель (ЦКМ) содержит грамматическую синтагму словоформы-омонима и ее контекстное окружение в виде синтагм словоформ контекстного окружения, а результатом разрешения омонимии является однозначный выбор конкретной словоформы-омонима. При разработке этой модели авторы исходили из следующей гипотезы: *одинаковым последовательностям обобщенных символов классов слов (обобщенным синтагмам) должны соответствовать одинаковые синтаксические структуры различных фрагментов текстов*. Кратко рассмотрим идею модели ЦКМ применительно к задаче разрешения омонимии.

3.1 Модель ЦКМ для разрешения омонимии

На вход ЦКМ подается упорядоченное множество объектов $\mathrm{Sem} = \{s_i\}_{i=1}^{n}$. Каждому объекту множества (синтагме) Sem ставится в соответствие элемент упорядоченного множества $\mathrm{Pr} = \{p_i\}_{i=1}^{n}, s_i \leftrightarrow p_i$, где p_i – свойства объекта.

Задается радиус n, выбирается целевой токен (центроид) s_k, составляется позиционная модель, которая задается упорядоченной последовательностью (рис. 1):

$$\mathrm{PM} = \{s_k, s_{k+1}, s_{k-1}, s_{k+2}, s_{k-2}, \dots, s_{k+n}, s_{k-n}\}. \quad (1)$$

Рис. 1. Позиционная модель
Fig. 1. Positional model

Заполнение позиционной модели показано на рис. 2.

Рис. 2. Заполнение позиционной модели
Fig. 2. Filling in the positional model

К упорядоченной последовательности PM применяется вектор-функция fPR, которая каждому токену из PM ставит в соответствие признаки из множества Pr:

$$\mathrm{fPR}(s_i) = \begin{cases} p_i, & s_i \in \mathrm{Sem}, \\ p_0, & s_i \notin \mathrm{Sem}. \end{cases}$$

Здесь p_0 – фиктивный признак фиктивного элемента. В результате применения функции получается упорядоченная последовательность:

$$\mathrm{fPR}(\mathrm{PM}) = \mathrm{PMp} = \{p_k, p_{k+1}, p_{k-1}, p_{k+2}, p_{k-2}, \dots, p_{k+n}, p_{k-n}\}.$$

Пусть существует функция fSim (similar function), которая проверяет, на сколько два вектора схожи между собой:

$$\mathrm{fSim}(a, b) = i, \text{если } \forall k \in \{1, \dots, i\}\ a_k = b_k \text{ и } a_{i+1} \neq b_{i+1}.$$

Задается порог совпадения начальной части модели bv и во множестве MOD шаблонов синтагм ЦКМ из словаря шаблонов ищется наиболее схожая структура, наиболее схожая в смысле максимума процента покрытия CovPer:

$$\mathrm{PMpMod} = \underset{m \in \mathrm{MOD}}{\operatorname{argmax}}\, \mathrm{CovPer}(\mathrm{PMp}, m) = \begin{cases} 0\%, & \mathrm{fSim}(\mathrm{PMp}, m) < bv \\ \frac{l - \mathrm{fHam}(\mathrm{PMp}, m)}{l} \cdot 100\%, & \mathrm{fSim}(\mathrm{PMp}, m) \geq bv' \end{cases}$$

где $\mathrm{fHam}(\cdot,\cdot)$ – функция, которая считает расстояние Хемминга (количество не совпавших элементов), $l = 2n + 1$ – количество элементов синтагмы.

Множество MOD содержит в себе шаблоны синтагм ЦКМ из словаря шаблонов. Каждый элемент $m \in \mathrm{MOD}$ имеет признак, который принимает значение 0 или 1.

Пусть функция ftakePR – функция взятия признака элемента $m \in \mathrm{MOD}$.

В результате ЦКМ содержится ответ на вопрос: *Является ли проверяемая структура той структурой, на соответствие которой мы ее проверяем или нет?* Ответ дается следующим образом:

$$\mathrm{answer} = \begin{cases} \text{да}, & \mathrm{ftakePR}(\mathrm{PMpMod}) = 1, \\ \text{нет}, & \mathrm{ftakePR}(\mathrm{PMpMod}) = 0. \end{cases}$$

3.2 Автоматизированное формирование словарей ЦКМ для разрешения омонимии

Необходимыми данными для формирования словарей ЦКМ по корпусу текстов служит массив омонимичных словоформ. В процессе его формирования выявляются предложения, в которых содержатся омонимичные словоформы и для каждой такой словоформы строится ЦКМ. В процессе обработки корпуса текстов устанавливаются наиболее частые формы омонимов (доминантные словоформы-омонимы). При этом пользователю предоставляется возможность корректировать грамматическую информацию словоформы-омонима, соответствующую конкретному контексту. Таким образом, создаются два словаря ЦКМ: словарь доминантных ЦКМ и словарь ЦКМ для редких (альтернативных) словоформ-омонимов. Но, поскольку требуется установить один из двух вариантов омонимов, необходимо определить: *является ли словоформа доминантной или не является,* то для этого достаточно только одного словаря доминантных словоформ. В табл.2 показан макет интерфейса для формирования и коррекции словаря ЦКМ для разрешения омонимии.

Табл. 2. Представление визуального интерфейса для ручного разрешения омонимии в ЦКМ
Tab. 2. Presentation of the visual interface for manual resolution of homonymy in the CCM

Перенумерованное исходное предложение №1:
00 Океанские || 01 суда || 02 стояли || 03 на || 04 рейде || 05 . ||

Исходное предложение, обработанное процедурой МА:
00 Океанские #OK=2#FK=106#GI=*0210*0240#GK=A#OS=Pg#TD=E
01 суда #OK=1#FK=070#GI=*1120#GK=N#OS=ГВ#CK=A#TD=S
02 стояли #OK=1#FK=125#GI=*0200#GK=L#OS=Яf#PG=t#CK=R#TD=S
03 на #OK=0#FK=164#GI=*0040*0060#GK=F#OS=ыA#CK=e#TD=S
04 рейде #OK=1#FK=001#GI=*1160#GK=N#OS=AK#TD=K
05 . #OK=0#FK=0#TW=0#GI=*0000#OS=..#GK=.#TD=Z

ГВЯfPgыAZZAKZZ..ZZZZZZ – доминантная ЦКМ для слова, имеющего ФК=001 и ОК=a)

Перенумерованное исходное предложение №2:
00 Заседание || 01 суда || 02 состоится || 03 в || 04 11 || 05 часов || 06 . ||

00 Заседание#OK=1#FK=073#GI=*3110*3140#GK=N#OS=ЁK#SU=t#CK=A#TD=S
01 суда #OK=1#FK=001#GI=*1120#GK=N#OS=AB#CK=A#TD=S
02 состоится#OK=4#FK=117#GI=*0103#GK=V#OS=Щs#PG=t#PV=t#CK=Q#TD=S
03 в #OK=0#FK=164#GI=*0040*0060#GK=F#OS=ыA#CK=e#TD=S
04 11 #OK=0#FK=313#GI=*0000#GK=0#OS=9A#SU=t#P0=t#TD=0
05 часов #OK=2#FK=001#GI=*1220#GK=N#OS=Az#CK=A#TD=S
06 . #OK=0#FK=0#TW=0#GI=*0000#OS=..#GK=.#TD=Z

ГВЯfPgыAZZAKZZ..ZZZZZZ – доминантная ЦКМ для слова, имеющего ФК=001 и ОК=a)

4. Разрешение омонимии на основе словарей ЦКМ

Под разрешением омонимии понимается установление конкретного набора грамматических и семантических характеристик омонимичной словоформы, соответствующее ее смысловому значению в конкретном контекстном окружении. При реализации данного метода в основном морфологическом словаре (SYS_MA_DCCM) были представлены все выявленные омонимичные словоформы с грамматическими признаками доминантной формы и с указанием признака омонима (PO=t). Альтернативные формы омонимов были представлены в словаре (SYS_MA_ACCM). Разрешение омонимии выполнялось по словарю ЦКМ (SYS_Dom_Hom). В случае полного или частичного совпадения текстовой ситуации и словарного элемента словаря SYS_Dom_Hom выбиралась доминантная форма омонима из словаря SYS_MA_DCCM, если такого совпадения не было, выбиралась альтернативна форма из словаря SYS_MA_ACCM.

Для определения критерия частичного совпадения эталонной модели ЦКМ и текстовой модели ЦКМ был проведен лингвистический анализ различных омонимичных ситуаций в корпусе текстов объемом 12 млн. словоформ. На основе этого анализа было выявлено и проанализировано 43 типа омонимичных трансформаций. Для каждого типа трансформаций был разработан механизм разрешения омонимии (МРО) на основе применения модели ЦКМ.

Табл. 3. Фрагмент таблицы типов омонимичных трансформаций омонимичных словоформ
Table 3. Fragment of the table of types of homonymous transformations of homonymous word forms

№ класса и типа омонимичной трансформации	**Грамматические трансформационные классы омонимов**	**Слово-представитель**	**Обобщенная синтагма № 1**	**Обобщенная синтагма № 2**	**Минимальное совпадение элементов ЦКМ (тест. и слов.)**
1. Местоименные существительные – местоименные прилагательные					
1.1	y-s	ее	0A	pA	3
1.2	y-s	его	яA	pA	3
1.3	y-s	их	4A	pA	3
2. Субстантивированные прилагательные-существительные					
2.1	A-N	легкое	РБ	РБ	6
2.2	A-N	сборная	HI	HI	6
3. Омонимичные существительные - прилагательные					
3.1	N-A	долгом	ГГ	ГГ	7
3.2	N-A	теплом	ГГ	ГГ	7
3.3	N-A	правом	ГГ	ГГ	7
…					
4. Существительные фамильно-именной группы (мужской род, им. падеж) - существительные					
4.1	N-N	быков	Yz	hA	6
4.2	N-N	волков	Yz	hA	6
…					
5. Существительные фамильно-именной группы (женский род, им. падеж) – существительные фамильно-именной группы (мужской род, косв. падеж)					
5.1	N-N	иванова	jB	hA	6
5.2	N-N	петрова	jB	hA	6

В табл. 4 приведены фрагменты типов омонимии и их доля в текстах, также вероятности разрешения этих типов на основе применения ЦКМ.

Табл. 4. Типы омонимии и их доля в текстах и вероятности разрешения этих типов на основе применения ЦКМ
Table 4. Types of homonymy and their share in texts and the probabilities of resolving these types based on the use of CCM

	Класс словоформ-омонимов	**Вероятность метода**	**Доля класса в массиве (тыс.)**	**Относительная доля класса в массиве (%)**
1	Местоименные существительные – местоименные прилагательные	**98%**	**727**	**49,0**
2	Субстантивированные прилагательные-существительные	**95%**	**17**	**1,14**

3	Омонимичные существительные - прилагательные	**95%**	**11**	**0,75**
4	Существительные фамильно-именной группы (мужской род, им. падеж) – существительные	**95%**	**16**	**1,07**
5	Существительные одушевленные (мужской род, им. падеж) - существительные неодушевленные	**95%**	**9**	**0,66**
6	Существительные неодушевленные (женский род, им. падеж)	**95%**	**1**	**0,09**
7	Существительные одушевленные муж. рода – существительные неодушевленные жен. рода	75%	16	1,09
8	Существительные неодушевленные жен. рода – существительные неодушевленные муж. рода	75%	57	4,05
9	Существительные одушевленные (женский род, им. падеж) - существительные неодушевленные	75%	6	0,03
10	Существительные одушевленные муж. рода – существительные неодушевленные муж. рода	75%	43	1,45
11	Существительные неодушевленные	75%	155	7,61
12	Глагол личной формы ед. числа 1-го лица	75%	19	1,32
13	Глагол личной формы мн. числа 2-го лица	75%	0,132	0,132
14	Глаголы повелительного наклонения – существительные	75%	0,67	0,67
15	Существительные - краткие причастия	75%	13	1,23

В этом фрагменте таблице (табл. 4) приводится доля классов омонимов от их общего числа и их абсолютное число в массиве, объемом 740 тыс. документов (12 млн. слов). В этом массиве обнаружено 1.5 млн. омонимичных случаев.

4.1 Алгоритм разрешения омонимии словоформ в текстах

Алгоритм разрешения омонимии выполняется на этапе семантико-синтаксического анализа на основе результатов обработки текста процедурой МА и состоит из следующих этапов.

Этап 1. В результатах анализа МА слов текста определяются слова по словарю SYS_MA_DCCM, в ГИ которых имеется признак омонимии (PO=t).

Этап 2. На основе грамматической информации омонимичной словоформы и грамматической информации слов, окружающих слева и справа от нее, строится ЦКМ.

Этап3. Если сформированная текстовая ЦКМ совпадает с одним из элементов словаря SYS_ Dom_Hom, а число совпавших синтагм текстовой ЦКМ и словарной ЦКМ больше или равно минимальному числу, указанному в словарной статье этой ЦКМ, то считается, что разрешение данного случая омонимии выполнено и эта словоформа является доминантной.

Этап 4. В случае, если сформированная ЦКМ не совпадает с одним из элементов словаря SYS_Dom_Hom, или число совпавших синтагм текстовой ЦКМ и словарной ЦКМ меньше или равно минимальному числу, указанному в словарной статье этой ЦКМ, то считается, что

разрешение данного случая омонимии выполнено и эта словоформа является альтернативной, а грамматическая информация назначается ей по словарю SYS_MA_ACCM.

5. Заключение

Предлагаемый метод выявления случаев омонимии в корпусе текстов и ее разрешения с помощью модели ЦКМ базируется на теоретической концепции фразеологического концептуального анализа текстов (ФКАТ) и уникальной машинной грамматике, в основу которой положена система флективных классов русских слов. Заложенное в теоретической концепции флективных классов слов русского языка жесткое соответствие между формой представления слов и их грамматической информацией позволило создать на этой основе новые классы – классы слов, имеющие одинаковые наборы грамматических признаков, соответствующие их формам представления в сходных контекстных окружениях. При разработке этой модели авторы исходили из следующей гипотезы: *одинаковым последовательностям обобщенных символов классов слов (обобщенным синтагмам) должны соответствовать одинаковые синтаксические структуры различных фрагментов текстов.*

В процессе выявления омонимичных словоформ применялись методы предиктивного назначения грамматических характеристик словоформам и формирования для всех словоформ их словоизменительных парадигм. Установление расхождений грамматических характеристик одинаковых словоформ позволило автоматически выявить омонимичные словоформы. Установление местоположений омонимичных словоформ в различных контекстных окружениях позволяет автоматически сформировать модели ЦКМ для разрешения омонимии. С помощью визуального интерфейса пользователь способен с минимальными трудозатратами откорректировать словари для разрешения омонимии. Основным достоинством метода является полная автоматизация процесса выявления омонимичных словоформ и высокая степень автоматизации создания декларативных средств для разрешения омонимии.

Разработанный авторами данной статьи новый метод автоматического выявления омонимичных словоформ в корпусе текстов и автоматизированное создание декларативных средств их разрешения позволяет быстро решать проблему разрешения омонимии для текстов любого лексического состава в различных тематических областях.

Список литературы / References

[1] Baum L.E., Petrie T. Statistical inference for probabilistic functions of finite state Markov chains. The annals of mathematical statistics, vol. 37, issue 6, 1966, pp. 1554-1563.

[2] Lafferty J., McCallum A., Pereira F.C.N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. of the Eighteenth International Conference on Machine Learning, 2001, pp. 282-289.

[3] Elman J.L. Finding structure in time. Cognitive science, vol. 14, issue 2, 1990, pp. 179-211.

[4] Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval, Cambridge University Press, 2008, 506 p.

[5] Sha F., Pereira F. Shallow parsing with conditional random fields. In Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, vol. 1, 2003, pp. 134-141.

[6] Хорошилов Александр А., Мусабаев Р.Р. и др. Автоматическое выявление и классификация информационных событий в текстах СМИ. Научно-техническая информация. Серия 2: Информационные процессы и системы, вып. 7, 2020 г., стр. 27-38 / Khoroshilov Alexandr A., Musabaev R.R. et al. Automatic Detection and Classification of Information Events in Media Texts. Automatic Documentation and Mathematical Linguistics, vol. 54, issue 4, 2020, pp. 202–214.

[7] Аблов И.В., Козичев В.Н и др. Средства машинной грамматики русского языка (по Г.Г. Белоногову). Научно-техническая информация. Серия 2: Информационные процессы и системы, вып. 6, 2018 г., стр. 32-46 / Ablov I.V., Kozichev V.N. The Tools of a Machine Grammar of the Russian

Language (based on G.G. Belonogov). Events in Media Texts. Automatic Documentation and Mathematical Linguistics, vol. 52, issue 3, 2020, pp. 142-156.

[8] Калинин Ю.П., Хорошилов Александр А., Хорошилов Алексей А. Современные технологии автоматизированной обработки текстовой информации. Системы высокой доступности, том 11, вып. 2, 2015 г. / Kalinin Yu.P., Khoroshilov Alexander A., Khoroshilov Alexey A. Modern technologies for automated text processing. High Availability Systems, vol. 11, issue 2, 2015 (in Russian).

Информация об авторах / Information about authors

Александр Алексеевич ХОРОШИЛОВ – доктор технических наук, профессор МАИ, ведущий научный сотрудник ФИЦ ИУ РАН, старший научный сотрудник 27 ЦНИИ Минобороны России. Область научных интересов: теоретические основы информатики; вычислительные машины электронные цифровые; машинный перевод; прикладное языкознание; автоматическая обработка речевой информации

Alexander Alexeevich KHOROSHILOV – Doctor of Science, Professor of the Moscow Aviation Institute (National Research University), Lead Researcher of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Senior Researcher of the 27 Central Research Institute of the Ministry of Defense of the Russian Federation. Research interests: theoretical foundations of informatics; electronic digital computers; Machine translate; applied linguistics; automatic processing of speech information

Юрий Викторович НИКИТИН – научный сотрудник ФИЦ ИУ РАН, руководитель группы разработки АО «НПК «ВТ и СС». Научные интересы: методы и технологии семантического анализа текстов, фразеологического машинного перевода и создания декларативных средств для лингвистического программного обеспечения.

Yuri Viktorovich NIKITIN – Researcher of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Development Team Leader of the Scientific and Industrial Company "High Technologies and Strategic Systems". Research interests: methods and technologies of semantic text analysis, phraseological machine translation and creation of declarative tools for linguistic software.

Анна Владимировна КАН – кандидат технических наук, доцент МАИ, начальник аналитического отдела ФГБУ «НИЦ «Институт имени Н.Е. Жуковского. Основные научные интересы в области системного анализа, имитационного моделирования и искусственного интеллекта.

Anna Vladimirovna KAN – Candidate of Technical Sciences, Associate Professor of the Moscow Aviation Institute, Head of the Analytical Department of the National Research Center "Zhukovsky Institute". Her main research interests are in the field of systems analysis, simulation and artificial intelligence.

Яна Дмитриевна КОЗЛОВСКАЯ – участник группы разработки АО «НПК «ВТ и СС». Научные интересы: компьютерная лингвистика, анализ текстов.

Yana Dmitrievna KOZLOVSKAYA – development team member of the Scientific and Industrial Company "High Technologies and Strategic Systems". Research interests: computational linguistics, text analysis.

Екатерина Андреевна ЕВДОКИМОВА – студентка. Научные интересы: анализ и обработка текстовых данных.

Ekaterina Andreevna EVDOKIMOVA – student of the Moscow Aviation Institute. Research interests: analysis and processing of text data.

DOI: 10.15514/ISPRAS-2022-34(5)-12

Overapproximation of the Number of Active Timers in Timed-Arc Petri Nets Using DP-Systems

L.W. Dworzanski, ORCID: 0000−0002−0074−7660 <leo@mathtech.ru>
National Research University Higher School of Economics,
Myasnitskaya st., 20, Moscow, 101000, Russia

Abstract. Timed-arcs Petri nets are a time extension of Petri nets that allows assigning clocks to tokens. System of dynamic points on a metric graph (DP-systems) is another dynamical model that is studied in discrete geometry dynamics; DP-system combines continuous time and discrete branching events and used, for example, in study of localized Gaussian wave packets scattering on thin structures. In recent works, asymptotic estimates of the growth of the number of points in dynamic systems on metric graphs were obtained. In this paper, we provide a mean to overapproximate the number of different values of timers for a subclass of timed-arc Petri nets by constructing a system of dynamic points on a metric graph and prove overapproximation of the number of timer values by the number of points in the system of dynamic points.

Keywords: metric graphs; timed-arc Petri nets; dynamic properties

For citation: Dworzanski L.W. Overapproximation of the Number of Active Timers in Timed-Arc Petri Nets Using DP-Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 183-194. DOI: 10.15514/ISPRAS-2022-34(5)-12

Acknowledgements. The reported study was funded by RFBR according to the research project 20-07-01103a.

Оценка сверху числа активных таймеров в сетях Петри с временными дугами с помощью динамических систем точек на графах

Л.В. Дворянский, ORCID: 0000−0002−0074−7660 <leo@mathtech.ru>
Национальный исследовательский университет «Высшая школа экономики»,
101000, Россия, Москва, Мясницкая ул., 20

Аннотация. Сети Петри с временными дугами – это временное расширение сетей Петри (TaPN-сети), которое позволяет присваивать таймеры фишкам. Система динамических точек на метрическом графе (DP-система) это другая динамическая модель, которая рассматривается в теории геометрических дискретных динамических систем и, исторически, ее изучение мотивировано изучением распространения локализованных гауссовых волновых пакетов по тонким структурам. DP-система моделирует дискретные ветвящиеся события происходящие в реальном времени. В недавних работах были получены асимптотические оценки на рост числа точек в DP-системах на метрических графах. В данной работе мы предлагаем методы оценки сверху числа различных значений таймеров для подкласса сетей Петри с временными дугами с помощью построения DP-системы на метрическом графе по сети Петри и показывает, что количество различных значений таймеров в исходной сети Петри не превосходят количество точек в DP-системе. Это позволяет переносить известные оценки для DP-систем на сети Петри с временными дугами для выделенного подкласса.

Ключевые слова: метрические графы; сети Петри с временными дугами; динамические свойства

Для цитирования: Дворянский Л.В. Оценка сверху числа активных таймеров в сетях Петри с временными дугами с помощью динамических систем точек на графах. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 183-194. DOI: 10.15514/ISPRAS-2022-34(5)-12

Благодарности: Работа выполнена при поддержке гранта РФФИ 20-07-01103 а.

1. Introduction

Petri nets are widely-used to model the behaviour of distributed concurrent computer systems and concurrent processes in biology, chemistry, physics, and other fields [1]. There are many time extensions of Petri nets were suggested [2, 3]. Simultaneously, almost any of semantical extensions makes Petri nets Turing-complete and many of general behavioral problems immediately become undecidable by Rice-Uspensky theorem. The widely known time extensions of Petri nets – Time Petri nets and Timed (Duration) Petri nets – are Turing-complete as they admit urgency and allow to model unbounded counters. Time extensions of Petri nets, as well as other real-time models, are under active study as, for many real-world software/hardware systems, time related aspects like performance, time-outs, delays, and latency are crucial for correct functioning [4-6]. A time semantics with restricted urgency was recently suggested for TaPN-nets in [7]; the suggested semantics allows urgent transitions to consume tokens only from the bounded places of a Petri net, and this restriction makes some behavioral problems decidable for TaPN-nets.

Timed-arc Petri nets (TaPN-nets) are an extension of Petri nets with real-time semantics: tokens are assigned clocks [8]; the inscription on an incoming arc of a transition define tokens of which age can be consumed by the firing of the transition.

TaPN-nets could be used to model mobile agents moving among nodes or data packets being transmitted between nodes of a telecommunication system. The number of tokens with different timers corresponds to the number of different data packets in a system. Examples of systems where the number of different packets is an important characteristic – networks of self-driven cars, high-loaded telecommunication systems experiencing DDoS attack, underwater network of drone swarm with intensive communication. Excessive number of packets stipulates communication quality degradation or failure. Therefore, to assess the quality of system functioning, it is important to estimate the number of tokens-packets with different timer values. In this paper, we suggest to an approach to approximate the number of different timers in a TaPN by the number of dynamic points on metric graphs.

Metric graph is a graph with lengths assigned to edges. Some results towards dynamical characteristics of systems of dynamic points flowing along undirected edges of a metric graphs in both directions were recently obtained [9-13]. In such systems, when a point reaches a vertex of the graph, new points start moving along all the edges incident to the vertex. The growth of the number of points moving along edges and its asymptotics were studied in [11, 12] and, for some special cases, in [14].

The orientation of metric graph edges allows us to approximate TaPN-nets more precisely; thus, in [15], the asymptotics of points number growth were obtained for systems of dynamic points on directed Sperner graphs. And later in [16], it was extended to directed Hamiltonian graphs.

The suggested simulation of a TaPN-net with a metric graph enables us to overapproximate the number of different clock values in the TaPN-net. A point in a constructed metric graph represents a clock value in the TaPN-net, and the event of the arrival of a point to a vertex simulates that the clock has reached a specific value. This allows us to estimate the number of different time events in a system modelled using TaPN-net. Translation to less expressive but more amenable to behavioural analysis or better studied models can be found in coverability analysis of Petri nets using Karp-Miller trees or in performance analysis of continuous timed Petri nets using Markov processes [17].

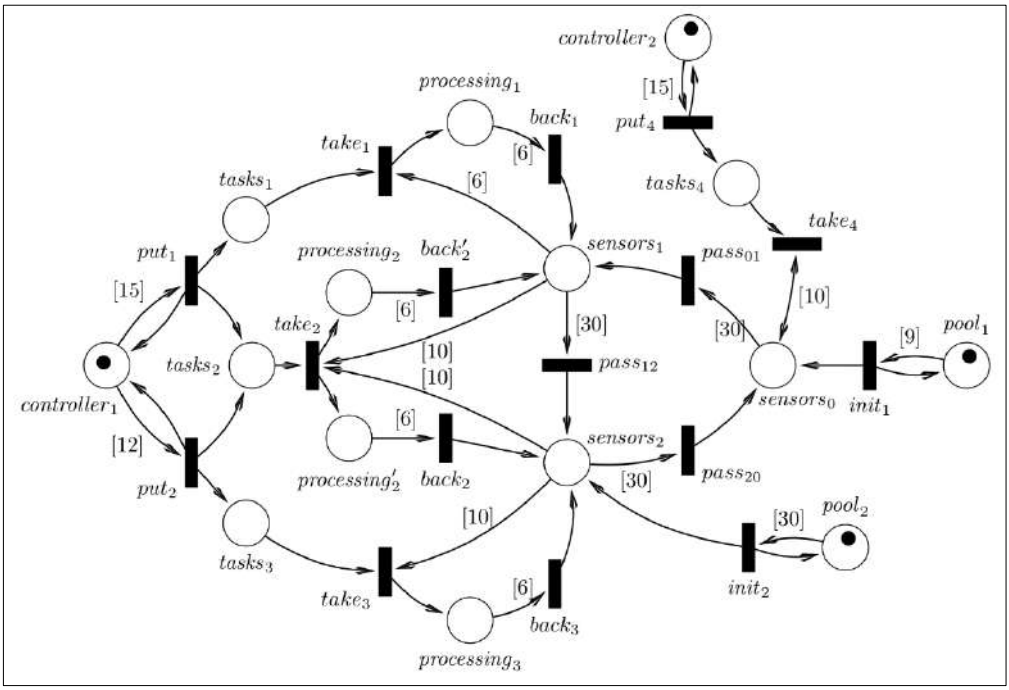


Fig. 1. TaPN-net model of an underwater wireless sensor network

The frequency of time events and its growth can be crucial as it determines energy or traffic load of the system. The growth of frequency can be important for such systems as software defined networks when resource management is adaptive [18] or when communication is complicated due to media. As a motivational example, we can consider a TaPN-net model of an underwater wireless sensor networks [19, 20]. The communication among underwater sensors is complicated due to the transmission medium (salt water); energy consumption and packets collisions impede efficient implementation of acoustic underwater transmission technology. Consider TaPN-net model $TaPN_{uwa}$ of an underwater acoustic sensor network (UWA-SN) in Figure 1. Places $sensors_{1\text{-}3}$ model spatial locations of underwater sensors, which measure temperature and pH levels. Sensors are regularly issued to places $sensors_0$ and $sensors_2$ from pools $pool_1$ and $pool_2$. Each 30 time units, a sensor flows to another spatial position. As communication is complicated and energy is an important resource, the active participation of sensors in communication shall be reduced to minimum. Thus, sensors receive commands to probe the water from controllers $controller_1$ and $controller_2$. Signal emission of controllers is modelled with places $tasks_{1\text{-}3}$. If a sensor is ready and receives a signal $tasks_{1\text{-}3}$, then it takes $take_{1\text{-}3}$ an assignment, processes $processing_{1-3}$ it, and returns $back_{1-3}$ to waiting states $sensors_{1-3}$.

A collision in a node may occur when several signals reach the node simultaneously. Such collisions depend on the spatial position of nodes in UWA-SN. When such a collision occurs, the node signals emitters about the problem, and the emitters cooperate to avoid collisions. Thus, the appearance of a new time event can result in additional traffic in the network. When many collisions occur, such reconfiguration can impede the normal functioning; therefore, the growth rate of the number of points could be an important parameter of the systems [19, 20].

To overapproximate the growth of different time event in the system TaPN , we build a metric graph Γ. The growth rate of points in Γ is not lower than thus of the number of different token-clock values in the TaPN-net. Thus, if the growth rate of Γ is acceptable, then so is the growth rate in TaPN.

Both models embrace real-time dynamics of discrete entities moving within a structure defined by a graph. We establish correspondence between a subclass of TaPN-nets and DP-systems. In Section

2, the notions of dynamical systems of points on metric graphs and timed-arc Petri nets are given. In Section 3, we provide the notion of overapproximation and supporting translation from a TaPN-net to a metric graph. In section 4, we provide a translation of a DP-system to a TaPN-net. Section 5 concludes the paper with some discussion on further research.

2. Preliminaries

By $\mathbb{N}$, $\mathbb{Q}_{\geq 0}$, $\mathbb{R}$, we denote the sets of natural, non-negative rational, and real numbers, respectively. The set of open and closed intervals over $\mathbb{R}_{\geq 0} \cup \{\infty\}$ is denoted by $\mathrm{I}(\mathbb{R}_{\geq 0})$. For a set S, a *bag* (*multiset*) m over S is a mapping $m : S \to \mathbb{N}$. The set of all bags over S is denoted by $\mathbb{N}^S$. We denote addition and subtraction of two bags by + and −, the number of all elements in m taking into account the multiplicity by $|m|$, and pointwise comparisons of bags by $=$, $<$, $>$, $\leq$, $\geq$, that are defined as $m_1 R m_2 \equiv \forall s \in S: m_1(s) R m_2(s)$ where R is one of $=, <, >, \leq, \geq$. We overload the set notation writing $\emptyset$ for the empty bag and $\in$ for the element inclusion.

2.1. DP-systems on directed metric graphs

A directed metric graph Γ is a graph consisting of set of vertices V, set of directed edges (arcs) E, and length function l mapping each arc $a = \langle v_1, v_2 \rangle \in E$ to a positive real, *i.e.*, $l: E \to \mathbb{R}_+$ [21].

The arc opposite to arc $a = \langle v_i, v_j \rangle$ is denoted by $\underline{a}$, *i.e.*, $\underline{a} = \langle v_j, v_i \rangle$. For two points x and y on the graph, metric $\rho(x, y)$ is the shortest distance between them, where distance is measured along the arcs of the graph additively. A walk is a finite or infinite sequence of arcs which joins a sequence of vertices.

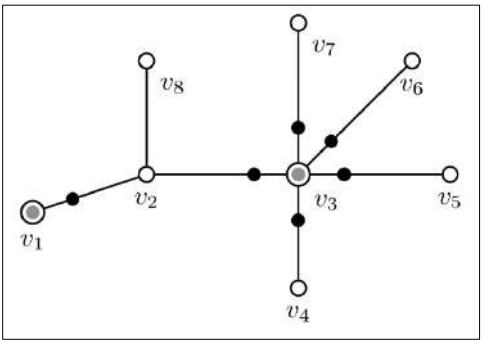


Fig. 2. System of dynamic points P_Γ on metric graph Γ

Definition 1 (System of dynamic points). *A system of dynamic points (DP-system) on a metric graph P_Γ is a pair $\langle \Gamma, P \rangle$, where Γ is metric graph and P is a set of points distributed on vertices and edges of Γ.*

Dynamics of a DP-system P_Γ is defined as following. In the initial state, a set of points is distributed on the arcs and vertices of Γ. The position of point p on arc a is denoted by $x_a(p)$ or just $x(p)$. When time starts to flow, each point p moves along its arc a direction. Each point p located at vertex v, for each outgoing arc a incident to v, produces a new point p_0 on each a, and p disappears (intuitively, this corresponds to wave packet scattering); each produced point p_0 starts moving along corresponding a. All points move with the same constant velocity; and, due to new points generation, some arcs may carry more than one point. When a moving point reaches the end incident to vertex v_0, again, on each outgoing arc incident to v_0, a new point is generated. When more than one points reach a vertex simultaneously at t, on each outgoing arc, only one point is produced, as if only one point has reached the vertex at t; *i.e.*, points met on a vertex fuse, and each coordinate of an arc can carry only one dynamic point.

In Figure 2, the initial set of points consists of two points in vertices v_1 and v_3. The point at v_1 produces a new point on edge $\langle v_1, v_2 \rangle$. The point at v_3 produces points on edges leading to vertices

v_2, v_4, v_5, v_6, v_7. After a time unit, there are no points in v_1 and v_3 (coloured grey), but there are points (coloured black) moving from v_1 and v_3 to their adjacent vertices.

The number of dynamic points in P_Γ at time t is denoted by $N_{P\Gamma}(t)$. The number of points on edge e at time t is denoted by $N_e(t)$. For dynamical systems of points P_Γ and $P'_{\Gamma'}$, we say that the growth rate of P_Γ is equal or less than thus of $P'_{\Gamma'}$ if $\forall t \in \mathbb{R}_+ \backslash Coll$: $N_{P\Gamma}(t) \le N_{P'\Gamma'}(t)$, where *Coll* is the (countable) set of time points when more than one dynamic point meet on a vertex; we exclude such time points as technically the number of dynamic points decreases for these moments. In what follows, we discuss number of points implicitly omitting vertices collision time points.

2.2. Timed-Arc Petri nets

Petri nets is a classical well-known formalism for concurrent systems modelling. A place/transition net (PT-net) is a Petri net with black tokens, that are indistinguishable from each other.

Definition 2 (Place/transition nets). *A PT-net is a tuple $\langle P,T,F,\gamma\rangle$, where*

- *P and T are disjoint finite sets of places, respectively, transitions;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);*
- *$\gamma : F \to N$ is a weight function.*

For an element $x \in P \cup T$, an arc $\langle y,x\rangle$ is called an *input arc*, and arc $\langle x,y\rangle$ – an *output arc*. *Preset* $^\bullet x$ and a *postset* $x^\bullet$ are subsets of $P \cup T$ such that $^\bullet x = \{y \mid \langle y,x\rangle \in F\}$ and $x^\bullet = \{y \mid \langle x,y\rangle \in F\}$. A *marking* of N is a function m: $P \to \mathbb{N}$. A pair $\langle N,m\rangle$ of a PT-net and a marking is called a marked net.

Let $N = \langle P,T,F,\gamma\rangle$ be a PT-net. A transition $t \in T$ is *enabled* in a marking m iff $\forall p \in {}^\bullet t \Rightarrow m(p) \ge \gamma\langle p,t\rangle$. An enabled transition t can *fire* yielding a new marking $m'(p) = m(p) - \gamma\langle p,t\rangle + \gamma\langle t,p\rangle$ for each $p \in P$ (denoted $m \to^t m'$). The set of all markings reachable from a marking m is denoted by $R(m)$.

Now, we provide the definition of Timed-Arc Petri nets (TaPN-nets) with token-based time semantics and urgency [6, 8, 22].

Definition 3 (Timed-Arc Petri net with Urgency). A TaPN-net is a tuple $TaPN = \langle N,\gamma^t,U\rangle$, where

- *$N = \langle P,T,F,\gamma\rangle$ is a PT-net called the skeleton of TaPN and denoted by S(TaPN);*
- *γ^t: $P \times T \to I(\mathbb{R}_+ \ge 0)$ is a set of token-age constraints on arcs;*
- *$U : T \to Q_{\ge 0}$ is a set of urgency constraints on transitions.*

The marking $m = \langle m_s,m_t,m_u\rangle$ of a TaPN-net *TaPN* consists of a marking m_s of $S(TaPN)$, a time marking m_t: $Tok(m_s) \to \mathbb{R}_{\ge 0}$ that assigns clocks to tokens, and an urgency marking m_U: $T(m_s) \to \mathbb{R}_{\ge 0}$ that assigns clocks to transitions, where $T(m_s)$ comprises all enabled transitions and $Tok(m_s)$ comprises all tokens of the marked PT-net $\langle S(TaPN),m_s\rangle$. The urgency constraint $U(t)$ means that t must fire if t has been enabled for $U(t)$ units of time. The token-age constraint $\gamma^t(p,t)$ defines that a firing of t may consume only token z in p with $m_t(z) \in \gamma^t(p,t)$. The urgency U of a transition is depicted as a number near the transition. The time constraints γ^t of an arc are depicted as an interval on the arc.

The operational semantics of TaPN-nets is defined by incorporating time constraints into the firing rules of PT-nets. Transition t is enabled in the marking $m = \langle m_s,m_t,m_u\rangle$, if t is enabled in $\langle S(TaPN),m_s\rangle$ and time constraints of t are satisfied, *i.e.*, each token α from a place p involved in the firing of t satisfies $m_t(z) \in \gamma^t(p,t)$.

An execution of a TaPN-net is a sequence of steps of the following two kinds.

A *transition firing step* is the firing of enabled transition t that consumes involved tokens from places $^\bullet t$ and produces new tokens to places $t^\bullet$; the urgency clock of t is set to zero, $m_u(t) = 0$, and, for each produced token α, the clock value is set to zero, *i.e.*, $m_t(\alpha) = 0$.

A *time elapsing* step corresponds to the elapsing δ time units in each clock of the marking m. We assume that all token clocks and transition clocks run at the same pace. We denote by $m+\delta$ the

marking with all its clocks increased by δ, *i.e.*, for each token $\alpha \in m_s$: $(m_t + \delta)(\alpha) = m_t(\alpha) + \delta$, and for each transition t : $(m_u + \delta)(t) = m_u(t) + \delta$. Under urgency restrictions, a time elapsing step δ is allowed if there are no $\delta' \in [0,\delta)$ such that the $m + \delta'$ marking has urgent transitions.

3. Overapproximation of the number of active timers

Coverability analysis or approximation using continuous Petri nets are classical examples of overapproximation in Petri net analysis. TaPN-nets with urgency are Turing-complete, and DP-systems on metric graphs are, obviously, less expressive (consider DP-systems dynamics monotonicity) [23]. Thus, we consider a restricted class of TaPN-nets, such that token-age constraints on input arcs are positive intervals of zero length and each transition t has zero urgency, *i.e.*, $U(t) = 0$.

In metric graphs, if points are met in a vertex simultaneously, they are coalesced, *i.e.*, the resulting effect is the same as if only one point came to the vertex. This peculiarity of metric graphs functioning hinders direct modelling of TaPN-nets using metric graphs. However, under the provided restrictions on TaPN nets, it is possible to overapproximate the number of different timers present simultaneously in a TaPN-net.

Let TaPN be a marked TaPN-net and $m = <m_s, m_t, m_u>$ be its marking. Let us introduce an equivalence relation $\approx_T$ over tokens:

$$\forall \alpha_1, \alpha_2 \in Tok(m): \alpha_1 \approx_T \alpha_2 \leftrightarrow \alpha_1 \in m_s(p) \wedge \alpha_2 \in m_s(p) \wedge m_t(\alpha_1) = m_t(\alpha_2)$$

where tokens α_1 and α_2 are $\approx_T$ -equivalent iff both are located in the same place p and have the same clock values.

When the clock value of a token α located in place p passes the maximum value of all the time constraints on outgoing arcs of p, it cannot further involve in firings and its clock becomes redundant. Such tokens are dead [16] by time restrictions and we may remove their clocks from consideration. We denote by *ActiveTokens(m)* the set of active tokens in marking m

$$ActiveTokens(m) = \{\alpha \in Tok(m) \mid \alpha \in m_s(p) \wedge m_t(\alpha) < max_{\forall <p,t> \in F}(\gamma_t <p, t>)\}$$

The quotient set $Timers(m) = ActiveTokens(m)/\approx_T$ consists of the classes of live tokens in marking m; each equivalence class corresponds to a set of tokens located in a place p that have the same clock value. In a physical system, a set of agents represented by such a class can share the same timer/external signal. The cardinality of $|Timers(m)|$ is the number of timers simultaneously active in the system.

Let $TaPN = <<P, T, F, \gamma>, \gamma^t, U>$ be a TaPN-net, where $P = \{p_1 ... p_n\}$ and $T = \{t_1 ... t_m\}$, and let m_0 be its initial marking. We construct DP-system $P_I(TaPN)$, which will be used to overapproximates *TaPN*, with simultaneously constructing relation $\varphi_b \subset Timers(m) \times P_I$ between timers *Timers(m)* and points in $P_I(TaPN)$ created to approximate these timers.

We start by adding a vertex v_i to $P_I(TaPN)$ for each transition $t_i \in T$. In Figure 3, an example of TaPN-net *TaPN* and $P_I(TaPN)$ that overapproximates *TaPN* is given. Then, for each pair of transitions t_i and t_j in T and place p_k in P such that there are edges $<t_i, p_k>$ and $<p_k, t_j>$ in F, we add an arc $<v_i, v_j>$ to $P_I(TaPN)$ with length $l(<v_i, v_j>) = \gamma_t(<p_k, t_j>)$. For each place p_k of P, for each token α in m_0 located in p_k, for each two transitions t_i and t_j, such that arcs $<t_i, p_k>$ and $<p_k, t_j>$ are in F, we put a new point p at the beginning of arc $<v_i, v_j>$ in $P_I(TaPN)$ and add $<[\alpha]_{\approx_T}, p_i>$ to φ_b. Technically, we don't put p to v_i but on $<v_i, v_j>$ as putting p to v_i would produce tokens to all outgoing arcs. For each marked place p_k in P that has only outgoing arcs, for each transition t_j in T adjacent to p_k, we introduce a new vertice v_0 and a new arc $<v_0, v_j>$ into $P_I(TaPN)$ with length $l(<v_0, v_j>) = \gamma^t(<p_i, t_j>)$; we put a point p to v_0 (as for p_1 and t_1 in Fig. 3) and add $<tm, p>$ to φ_b, where $tm = <p_i, 0>$. The intention is that each potential firing of a transition t_j in *TaPN*, producing tokens to its output places, corresponds to an event when a point in $P_I(TaPN)$ reaches v_j, generating points on its outgoing arcs. In Figure 3, two active tokens oscillate between places p_2 and p_3 in *TaPN*, while their corresponding points in $P_I(TaPN)$ oscillate on edges between v_2 and v_3.

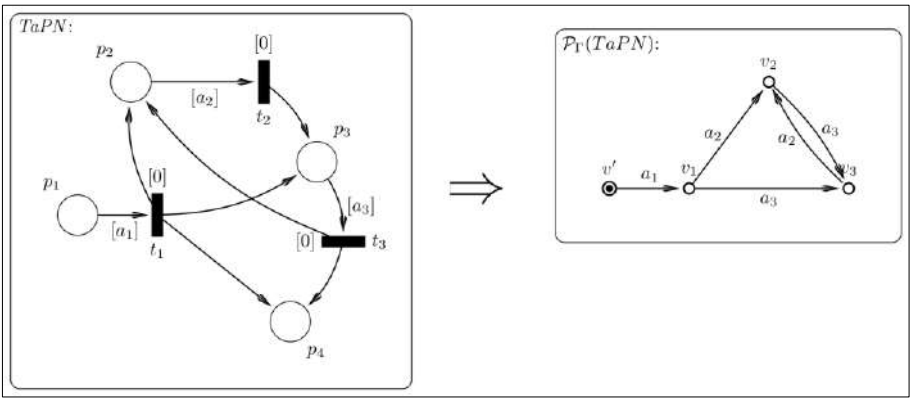


Fig. 2. TaPN-net TaPN and corresponding DP-system $P_\Gamma(TaPN)$

We say that P_Γ *overapproximates* TaPN-net *TaPN*, if for any *TaPN* execution $\sigma = m_0 \to^{s1} m_1 \to^{s2} m_2 \ldots$, for any state $m_i \in \sigma$ and global time $t \in \mathbb{R}_+ \backslash Coll$ that corresponds to m_i, $|Timers(m_i)| \le N_{P\Gamma}(t)$ holds.

Lemma 1. *Let TaPN be a TaPN-net with intervals of zero length as constraints on input arcs and each transition t has zero urgency. Let $P_\Gamma(TaPN)$ be a DP system constructed from TaPN. Then $P\Gamma(TaPN)$ overapproximates TaPN-net TaPN.*

Let $\sigma = m_0 \to^{s1} m_1 \to^{s2} m_2 \ldots$ be an execution of *TaPN*. For each m_i and corresponding time t, we define relation $\varphi_\tau \subset Timers(m_i) \times P_\Gamma$. For initial marking m_0 and time point $\tau = 0$, φ_0 is equal to φ_b. Let φ_t is defined for m_{i-1} and τ_{i-1} is global time corresponding to m_{i-1}. Let s_i be a time elapsing step for δ time units, and $\tau_i = \tau_{i-1} + \delta$ is the time corresponding to m_i.

For each place p_k, for each token $\alpha \in m_{i-1}$ located in p_k, for each two transitions t_l and t_j such that there are edges $<t_l, p_k>$ and $<p_k, t_j>$, if $(m_{i-1} + \delta)_t(\alpha) \le \gamma_t(<p_k, t_j>)$, add $<t_m, p>$ to $\varphi_{\tau i}$, where point p is located on edge $<v_l, v_j>$ at coordinate $x(p) = (m_{i-1} + \delta)_t(\alpha)$ and $t_m = <p_k, (m_{i-1} + \delta)_t(\alpha)>$. If $(m_{i-1} + \delta)_t(\alpha) > \gamma_t(<p_k, t_j>)$, we don't add anything to φ_τ as it means that the age of α won't let it be involved in a firing of t_j.

Let s_i be a transition firing step for transition t_j and $\tau_i = \tau_{i-1}$ as time doesn't flow during a transition firing step. All pairs related to all tokens consumed by the firing are not more in φ_τ. For each new token $\alpha \in m_i$ in place p_k produced by the t_j-firing for each two transitions t_l and t_q such that there are edges $<t_l, p_k>$ and $<p_k, t_q>$, we add $<tm, p>$ to $\varphi_{\tau i}$, where point p located at the beginning of edge $<v_l, v_q>$, and $tm = <p_k, 0>$.

Note that for each marking m_i, for each active token α, image $\varphi_\tau([\alpha]_{\approx T})$ is not empty as, when all the pairs related to α cease from φ_τ, α is dead and $[\alpha]_{\approx T}$ ceases from $Timers(m_i)$ by definition. For different equivalence classes $tm_1 = <p_1, \tau_1>$ and $tm_2 = <p_2, \tau_2>$, images $\varphi_\tau(tm_1)$ and $\varphi_\tau(tm_2)$ are disjoint as they either lie on different graph edges if $p_1 \ne p_2$, or lie on different coordinates if $\tau_1 \ne \tau_2$. Hence, $|Timers(m_i)| \le N_{P\Gamma}(\tau)$.

Now let us consider a class of TaPN-nets without urgency and with inscriptions $[x, \infty)$ on input arcs of transitions. Now a token α may be involved in a firing of transition t, even if its clock passed maximal interval lower bounds at time τ and t is disabled. When some other token α_0 comes to an input place p of t and α_0 clock satisfies inscription on arc $<p, t>$, transition t may become enabled and may fire consuming α. Such a token α is not dead at τ but we call it passive. For such TaPN-nets, set of active tokens *ActiveTokens(m)* is defined as follows

$$ActiveTokens(m) = \{\alpha \in Tok(m) \mid \alpha \in m_s(p) \wedge m_t(\alpha) < max\{x \mid <p,t> \in F \ \& \ \gamma_t<p,t> = [x,\infty)\}\}$$

Theorem 2. *Let TaPN be a TaPN-net without urgency, with inscriptions $[x, \infty)$ on input arcs of transitions, and with initial marking m_0. Let $P_\Gamma(TaPN)$ be a DP-system constructed from TaPN. Then $P_\Gamma(TaPN)$ overapproximates TaPN-net TaPN.*

Let t be a transition and its firing produces tokens $\alpha_1 \ldots \alpha_n$ to output places of t. As inscriptions are unbounded, t may fire not necessary immediately at its earliest enabling time τ but later, at time $\tau_0 = \tau + \delta$. Due to the monotonicity of Petri net firing rule and monotonicity of time constraints of form $[x, \infty)$, we may fire transition t at time τ and then just keep (freeze) tokens $\alpha_1 \ldots \alpha_n$ at output positions of t until $\tau + \delta$. Such execution may only increase the number of tokens in *TaPN* at interval $[\tau, \tau + \delta]$. Henceforth, when we want to overapproximate execution σ we may consider execution σ' such that if a transition t will fire then it will fire at its earliest enabling time.

Now we may apply the similar argument as in Lemma 1, and obtain $|Timers(m_i)| \leq N_{PI}(\tau)$.

As urgency may only narrow a set of possible executions of a TaPN-net, we get the following corollary.

Corollary 3. *Let TaPN be a TaPN-net with urgency, with inscriptions $[x, \infty)$ on input arcs of transitions, and with initial marking m_0. Let $P_I(TaPN)$ be a DP-system constructed from TaPN. Then $P_I(TaPN)$ overapproximates TaPN net TaPN.*

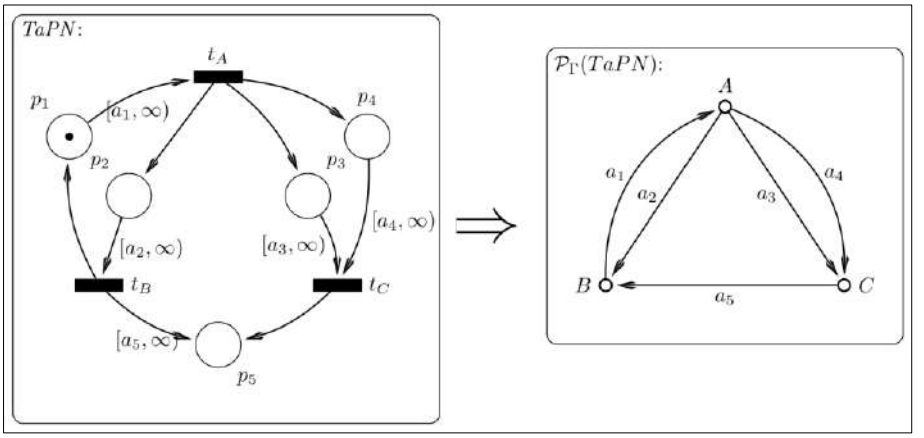


Fig. 4. TaPN-net TaPN and corresponding DP-system $P_I(TaPN)$

In Fig. 4, TaPN-net *TaPN* is on the left, and DP-system $P_I(TaPN)$ that overapproximates *TaPN* is on the right. In [7], it was calculated that the growth of the number of points in $P_I(TaPN)$ is

$$N(P_I(TaPN)) = t^2 (a_1 + a_2 + a_3 + a_4 + a_5) / (2(a_1 + a_4 + a_5)(a_2 + a_3)(a_1 + a_3 + a_5)) + O(t)$$

This gives us asymptotical estimate on the upper bound of the growth of number of active timers in TaPN.

In $P_I(TaPN)$, the arrival of a point to a vertex always corresponds to emitting of new points on the outgoing edges. In *TaPN*, the firing of transition t in *TaPN* may occur only when all input places $\bullet t$ have tokens with suitable clock values. Thus, some arrivals of points in Γ do not represent real firings of transition t in TaPN. In addition, tokens do not collapse in TaPN-nets; thus, we overapproximate the number of different clock values in TaPN. However, the number of different clock values is an important parameter for a system when, for example, its components use shared clocks.

4. Simulation of DP-systems using timed-arc Petri nets

To simulate DP-systems using Petri nets with real-time semantics, we need to represent points moving along edges of a graph using clocks in a Petri net. The advance of a point along an edge in a metric graph is modelled with the progress of the corresponding clock in a Petri net.

The number of points on a metric graph may grow indefinitely when edge lengths are incommensurable; therefore, it is not possible to model the evolution of a system of points on a metric graph using clocks in classical time or timed Petri nets [2, 24] because these models have finite structurally-determined number of clocks. On the contrary, in timed-arc Petri nets, clocks are

assigned to tokens [25], and the number of clocks can grow along with the number of tokens in the net; therefore, TaPN-nets with token-based time semantics suits well for simulating DP-systems.

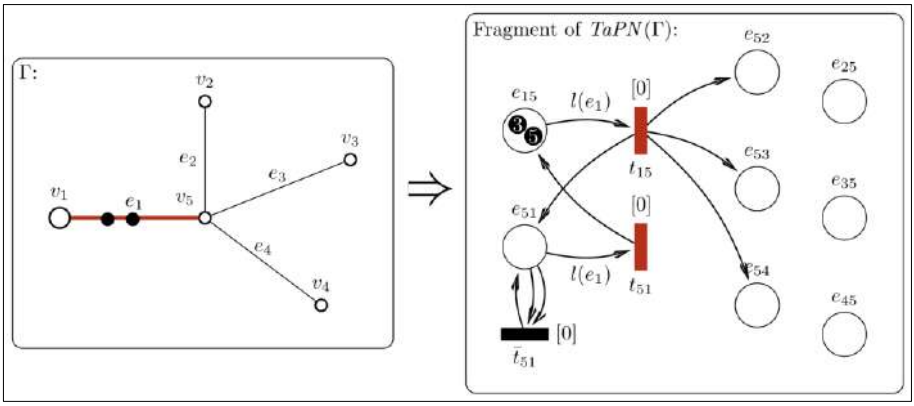


Fig. 5. A metric graph Γ and a fragment of TaPN(Γ) built from edge e_1

Let P_Γ be a DP-system and $TaPN(P_\Gamma)$ be the resultant TaPN-net simulating P_Γ. For each edge e of Γ connecting vertices a and b, we add two places e_{ab} and e_{ba}, where e_{ab} models arc $<a, b>$ and e_{ba} models arc $<b, a>$. In Fig. 5, DP-system P_Γ is on the left, and a fragment of its simulation $TaPN(P\Gamma)$ built for edge e_1 of Γ is on the right. Each point p on edge $<a, b>$ of P_Γ such that $x(p) = x_0$ is represented with a token α in p_{ab} such that $m_t(\alpha) = x_0$. In Fig. 5, two points on edge $<v_1, v_5>$ at the distances 3 and 5 from v_1 are modelled with two tokens that have timer values of 3 and 5, correspondingly. In addition, we add transitions t_{ab} and t_{ba} with $U(t_{ab}) = U(t_{ba}) = 0$, which firings model events when a point on e reaches b and a, respectively. We add arc $<e_{ab}, t_{ab}>$ with $\gamma t(<e_{ab}, t_{ab}>) = l(e)$, and, for each edge e_i in the metric graph connecting vertex b with vertex c, we add arc $<t_{ab}, e_{bc}>$ with $\gamma t(<t_{ab}, e_{bc}>) = 0$. Also, for each place e_{ab}, we add transition $\underline{t^{ab}}$ with $U(t^{ab}) = 0$ (as for e_{51} in Fig. 5); t^{ab} models collapsing of points in a vertex by consuming two tokens with clocks equals to 0 in e_{ab}, and puts only one token with a clock equal to 0 back.

The tool support and details of the translation software implementation are provided in [26]. The translation enables us to utilise well-known analysis tools for TaPN nets – TAPAAL/UPPAAL[29] and ReNew [30], to conduct behaviour analysis and numerical experiments for metric graphs. In Fig. 6, the translation of a metric graph DP to TaPN-net in TAPAAL representation using the developed tool is demonstrated.

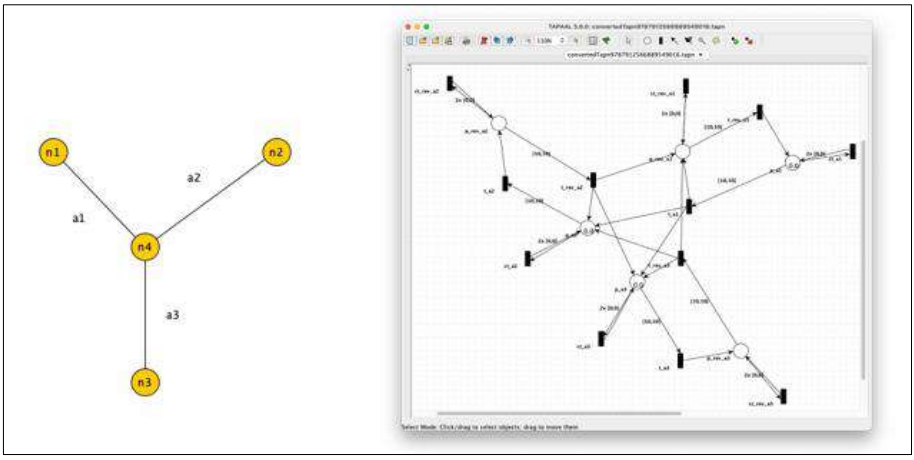


Fig.6. Tool support for translation of metric graph to TaPN-net

For example, the following properties can be checked using TCTL engine of TAPAAL:

– is it possible that more than 50 signals come to node X in 10 seconds;
– is it possible that in the next 5 minutes two messages come with the difference of their time moments less that ε milliseconds;
– does it hold, for the system initial phase with duration of 15 minutes, that if a point comes to node X, then no points comes to node Y within 10 seconds,

etc.

5. Conclusion

In this paper, we studied correspondence between metric graphs and a restricted class of TaPN-nets. We demonstrated that, for a restricted subclass of TaPN nets, it is possible to overapproximate the number of active timers using DP system on a metric graph. Such a correspondence connects studies on TaPN-nets and metric graphs, which allows to conduct analysis of the number of different timers in a given subclass of TaPN-nets using a system of points dynamics.

Our current research is to further weaken the suggested restrictions on TaPN-nets and, simultaneously, to find asymptotics for a more general class of DP-systems. Current working hypothesis is that the approach used to obtain estimates in [15, 16] could be extended to arbitrary directed graphs letting the suggested overapproximation to be used for a wider class of TaPN-nets. Conducted computed numerical experiments support it.

References / Список литературы

[1] Reisig W. Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013, 257 p.
[2] Berard B., Cassez F. et al. Comparison of different semantics for time Petri Nets. Lecture Notes in Computer Science, vol. 3707, 2005, pp. 293-307.
[3] Brown C., Gurr D. Timing Petri Nets categorically. Lecture Notes in Computer Science, vol. 623, 1992, pp. 571-582.
[4] Tvardovskii A.S., Yevtushenko N.V. On reduced forms of initialized Finite State Machines with timeouts. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 2, 2020. pp. 125-134. DOI: 10.15514/ISPRAS-2020-32(2)-10.
[5] Tvardovskii A.S., Laputenko A.V. On the possibilities of FSM description of parallel composition of timed Finite State Machines. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 25-40 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-2 / Твардовский А.С., Лапутенко А.В. О возможностях автоматного описания параллельной композиции временных автоматов. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 25-40.
[6] Akshay S., Genest B., Hélouët L. Decidable Classes of Unbounded Petri Nets with Time and Urgency. Lecture Notes in Computer Science, vol. 9698, 2016, pp. 301–322.
[7] Hanisch H.-M. Analysis of place/transition nets with timed arcs and its application to batch process control. Lecture Notes in Computer Science, vol. 961, 1993, pp. 282–299.
[8] Chernyshev V., Shafarevich A. Statistics of Gaussian packets on metric and decorated graphs, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 372, 2014, article id 20130145.
[9] Dworzanski L.W. Towards dynamic-point systems on metric graphs with longest stabilization time. arXiv preprint arXiv:2010.12528, 15 p.
[10] Chernyshev V.L. Tolchennikov A.A., Correction to the leading term of asymptotics in the problem of counting the number of points moving on a metric tree. Russian Journal of Mathematical Physics, vol. 24, issue 3, 2017, pp. 290–298.
[11] Chernyshev V.L., Tolchennikov A.A. The second term in the asymptotics for the number of points moving along a metric graph. Regular and Chaotic Dynamics, vol. 22, issue 8, 2017, pp. 937–948.
[12] Tolchennikov A.A., Chernyshev V.L., Shafarevich A.I. Asymptotic properties of the classical dynamical systems in quantum problems on singular spaces. Nonlinear Dynamics, vol. 6, issue 3, 2010, pp. 623-638 (in Russian)/ Толченников А.А., Чернышев В.Л., Шафаревич А.И. Асимптотические свойства и классические динамические системы в квантовых задачах на сингулярных пространствах. Нелинейная динамика, том 6, вып. 3, 2010 г., pp. 623-638.

[13] Chernyshev V. L., Tolchennikov A. A., Shafarevich A. I., Behavior of quasi-particles on hybrid spaces. relations to the geometry of geodesics and to the problems of analytic number theory, Regular and Chaotic Dynamics, vol. 21, issue 5, 2016, pp. 531-537.
[14] Chernyshev V., Tolchennikov A. Asymptotics of the number of endpoints of a random walk on a certain class of directed metric graphs. Russian Journal of Mathematical Physics, vol. 28, issue 4, 2021, pp. 434-438.
[15] Pyatko D., Chernyshev V. Asymptotics of the number of possible endpoints of a random walk on a directed hamiltonian metric graph. arXiv preprint arXiv:2112.13822, 2021, 15 p.
[16] Cohen G., Gaubert S., Quadrat J.-P., Asymptotic throughput of continuous timed petri nets. In Proc. of the 34th IEEE Conference on Decision and Control, vol. 2, 1995, pp. 2029-2034.
[17] Tuncer D., Charalambides M. et al. Adaptive resource man agement and control in software defined networks, IEEE Transactions on Network and Service Management, vol. 12, issue 1, 2015, pp. 18-33.
[18] Heidemann J., Stojanovic M., Zorzi M. Underwater sensor networks: applications, advances and challenges. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 370, 2012, pp. 158-175.
[19] Liu K., Yang Z. et al. Oceansense: monitoring the sea with wireless sensor networks. Mobile Computing and Communications Review, vol. 14, issue 2, 2010, pp. 7-9.
[20] Berkolaiko G., Kuchment P. Introduction to quantum graphs. Mathematical Surveys and Monographs, vol. 186. American Mathematical Society, 2013, 270 p.
[21] Bolognesi T., Lucidi F., Trigila S. From timed Petri Nets to timed LOTOS. In Proc. of the IFIP WG6.1 Tenth International Symposium on Protocol Specification, Testing and Verification X, 1990, pp. 377-406.
[22] de Frutos-Escrig D., Ruiz V.V., Alonso O.M. Decidability of properties of timed-arc petri nets. Lecture Notes in Computer Science, vol. 1825, 2000, pp. 187–206.
[23] Merlin P. A study of the recoverability of computer systems. Ph.D. Thesis, University of California, Irvine, CA, 1974, 154 p.
[24] Izmaylov A.A., Dworzanski L.W. Analysis of DP-systems Using Timed-Arc Petri Nets via TAPAAL Tool. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 155-166. DOI: 10.15514/ISPRAS-2020-32(6)-12.
[25] David A., Jacobsen L. et al. TAPAAL 2.0: Integrated Development Environment for Timed-Arc Petri Nets. Lecture Notes in Computer Science, vol. 7214, 2012, pp. 492-497.
[26] Cabac L., Haustermann M., Mosteller D. Renew 2.5 – towards a comprehensive integrated development environment for Petri Net-based applications. Lecture Notes in Computer Science, vol. 9698, 2016, pp. 101–112.

Information about the author / Информация об авторе

Leonid Wladimirovich DWORZANSKI – doctor of natural sciences (Doctor rerum naturalium), independent researcher. Research interests: analysis methods for discrete event dynamical systems, models for parallel and distributed computation, models for real-time systems, well-structured transition systems.

Леонид Владимирович ДВОРЯНСКИЙ – доктор естественных наук (Doctor rerum naturalium), независимый исследователь. Сфера научных интересов: методы анализа динамических свойств дискретных динамических систем, моделей параллельных и распределенных систем, моделей систем реального времени, вполне структурированных систем переходов.

DOI: 10.15514/ISPRAS-2022-34(5)-13

Турбулентные течения газа в каналах различных форм поперечного сечения с массоподводом

Б.Я. Бендерский, ORCID: 0000-0002-5482-3307 <bib@istu.ru>
А.А. Чернова, ORCID: 0000-0001-8579-6279 <alicaaa@gmail.com>
Ижевский государственный технический университет имени М.Т. Калашникова,
426069, Россия, г. Ижевск, ул. Студенческая, д. 7

Аннотация. Рассматриваются вопросы математического моделирования турбулентных течений в каналах со вдувом различных форм поперечного сечения. В результате серии вычислительных экспериментов с использованием инструментов пакета OpenFoam исследуются вопросы влияния формы канала на реализуемые особенности течений. Предложена математическая модель сопряженного теплообмена для рассматриваемых каналов. Сравнение результатов численного моделирования с известными экспериментальными данными показало корректность предложенных моделей, схем и алгоритмов. Исследованы топологические особенности структуры потока вязкого сжимаемого теплопроводного газа в каналах с массоподводом сложных форм, в том числе описаны особенности формируемых профилей скоростей в выходных сечениях каналов, приводятся результаты расчета коэффициента неравномерности скорости.

Ключевые слова: математическое моделирование; газовая динамика; прямолинейные каналы; массоподвод; профиль скорости

Для цитирования: Бендерский Б.Я., Чернова А.А. Турбулентные течения газа в каналах различных форм поперечного сечения с массоподводом. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 195-204. DOI: 10.15514/ISPRAS-2022-34(5)-13

Turbulent gas flows in channels of different cross-sectional shapes with mass flow

B.Ya. Benderskyi, ORCID: 0000-0002-5482-3307 <bib@istu.ru>
A.A. Chernova, ORCID: 0000-0001-8579-6279 <alicaaa@gmail.com>
Kalashnikov Izhevsk State Technical University
7, Studencheskaya st., Izhevsk, 426000, Russia

Abstract. The issues of mathematical modeling of turbulent flows in channels with blowing of different cross-sectional shapes are considered. As a result of a series of computational experiments using the OpenFoam tools, the influence of the channel shape on the realized features of flows is investigated. A mathematical model of conjugate heat transfer for the channels under consideration is proposed. Comparison of the results of numerical simulation with the known experimental data has shown the correctness of the proposed models, schemes and algorithms. The topological features of the structure of viscous compressible heat-conducting gas flow in the channels with mass flow of complex shapes have been studied, including the features of the velocity profiles formed in the outlet sections of the channels, the results of calculating the coefficient of non-uniformity of velocity have been described.

Keywords: mathematical modeling; gas dynamics; rectilinear channels; mass flow; velocity profile

For citation: Benderskyi B.Ya., Chernova A.A. Turbulent gas flows in channels of different cross-sectional shapes with mass flow. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 195-204 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-13

1. Введение

Каналы различных форм поперечного сечения широко используются в различных технических и природных системах. Поэтому внимание исследователей к вопросам организации течений жидкостей и газов в этих каналах не ослабевает.

Как правило канал представляет собой прямолинейный участок с произвольной формой поперечного сечения. В силу геометрических особенностей, а именно, явного преобладания линейных размеров (длины) над радиальными, наиболее часто исследования канальных течений проводятся в осесимметричной или плоской [1-3] постановках. В общем случае [2, 4-6] течения жидкостей и газов в таких каналах подробно исследованы как экспериментально, так и численно [7]. Отдельный интерес и потенциальную сложность для исследования представляют течения в проницаемых каналах и в каналах с подвижными границами вдува.

Проницаемость стенок канала, как и осуществление вдува с внутренних поверхностей каналов, согласно [4,5] приводит к формированию одиночных струек с поверхности массоподвода, с одной стороны, и требует обязательного учета трехмерности реализуемых течений при их численном моделировании [6-9], с другой. При этом взаимодействие единичных струек вблизи поверхности вдува приводит к дополнительной турбулизации потока и изменению режима течения. Совокупность вышеприведенных особенностей течений в каналах с проницаемыми стенками обуславливает повышение сложности численного моделирования процессов внутренней аэрогидрогазодинамики и приводит к увеличению числа факторов, оказывающих влияние на результат в физическом эксперименте. В силу этих причин особенности течений газа и жидкости в каналах с массоподводом остаются исследованными не полностью. Таким образом, работа посвящена математическому моделированию внутренней газодинамики в прямолинейных каналах сложной формы поперечного сечения с массоподводом.

2. Математическая постановка

Пространственное турбулентное течение сжимаемого теплопроводного газа в каналах с массоподводом описывается системой уравнений:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0; \tag{1}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right); \tag{2}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho E u_j}{\partial x_j} = -\frac{\partial p u_j}{\partial x_j} + \frac{\partial u_i \tau_{ij}}{\partial x_j} + \frac{\partial q_j}{\partial x_j}; \tag{3}$$

$$c\rho\frac{\partial T}{\partial t} = \lambda\nabla^2 T; \tag{4}$$

$$p = \rho R T, \tag{5}$$

где ρ – плотность газа, p – давление, u_i – компоненты вектора скорости, T – температура, $q_j = \lambda\frac{\partial T}{\partial x_j}$ – тепловой поток, R – удельная газовая постоянная; μ – динамический коэффициент вязкости; λ – коэффициент теплопроводности среды, $E = c_v T + 0.5u_i^2$ – полная удельная энергия среды, $\tau_{ij} = 2\mu S_{ij} - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}$ – тензор вязких напряжений; $S_{ij} =$

$\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)$ – тензор скоростей деформаций. Рассматривается квазистационарная постановка.

Поскольку в данной работе рассматриваются течения сжимаемого газа в каналах с массоподводом и сложной формой поперечного сечения, вследствие осуществления вдува газа с поверхностей массоподвода [5,10] возможно взаимодействие разнонаправленных струй газа, что обуславливает турбулизацию течений. Поэтому при моделировании внутренних канальных течений, даже при небольших скоростях вдува и числах Рейнольдса $10^3 \div 10^5$, необходимо учитывать турбулентность реализуемых течений. Таким образом, исходная система уравнений (1) – (5) согласно работе [11] осредняется по Фавру и Рейнольдсу, что позволяет учесть как турбулентные колебания, так и колебания плотности. Для замыкания полученной системы уравнений с учетом представленных в [12] результатов верификации моделей турбулентности применяется модель переноса компонент напряжения Рейнольдса в рамках модели Mentor SST [13-16]. Все константы модели рассчитаны в соответствии [11, 16-18] со значениями из стандартных моделей $k-\varepsilon$ и $k-\omega$ и представлены в [15, 19].

Рабочее тело – воздух ($M = 0.02896$ кг/моль, $c_p = 1004.4$ Дж/кгК, $\mu = 1.831 \cdot 10^{-5}$~кг/мс, $\lambda = 0.0261$ Вт/мК). Граничные условия приведены в табл. 1.

Табл. 1. Граничные условия
Table 1. Boundary conditions

Граница	**Граничное условие**
Граница вдува Г1	$p_0 = 5 \div 100 \cdot 10^5$Па, $T_0 = 600K$
Выход из канала Г2	$\frac{\partial p}{\partial n} = 0$
Стенки каналов	$u_i = 0, \frac{\partial u_i}{\partial \mathbf{n}} = 0, -\lambda_w \frac{\partial T_w}{\partial \mathbf{n}} = -\lambda_g \frac{\partial T_g}{\partial \mathbf{n}},\ T_w = T_g$

Расчетные сетки построены с использованием утилит построения сеток blockMesh и topoSet пакета openFoam и содержат более 5 млн призматических элементов с контролем положения первой ячейки в пристеночном слое ($y+ = 1 \div 10$), вопросы исследования сеточной сходимости рассмотрены в работе [15]. Сетки на границах сопряжения gas/solid являются согласованными.

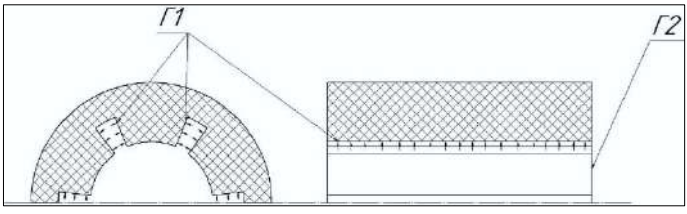


Рис. 1. Расчетная область
Fig. 1. Calculation area

Рассмотрим особенности течения газа в каналах с массоподводом различных форм поперечного сечения. Пример построения расчетной области для звездообразного канала приведен на рис. 1.

3. Анализ результатов численного моделирования

Течения газа в канале звездообразной формы характеризуется образованием парных вихрей (рис.2) в межщелевой части цилиндрического канала. Как показано в [20] к образованию парных вихревых структур приводит взаимодействие разнонаправленных струй газа, поступающих из лучей канала и с цилиндрической поверхности канала между собой.

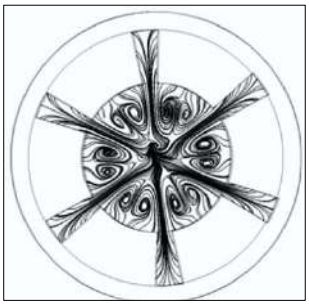
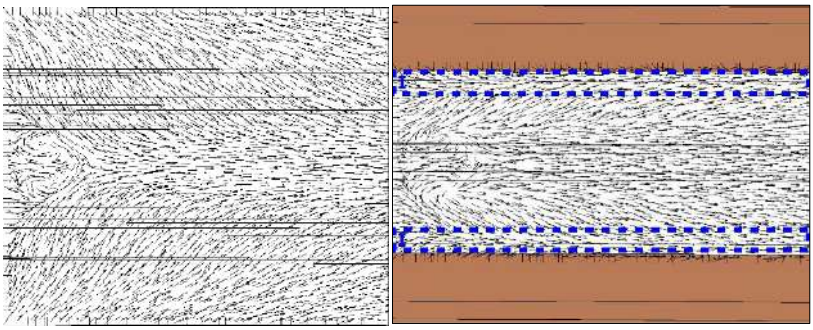


Рис. 2. Структура потока в выходном сечении проницаемого канала звездообразной формы
Fig. 2. Flow structure in the outlet section of the permeable star-shaped channel

На векторных картинах течения (рис. 3), в сечениях по лучам фиксируется формирование на входе в цилиндрическую часть канала области возвратных течений (рис. 3,а). Также наблюдается образование зон смешения потоков в поперечном сечении по лучам звезды (рис. 3,а) и вблизи цилиндрической стенки канала, в области между лучами (рис.3,б). Локальные особенности потока, фиксируемые на векторных картинах скоростей обусловлены формированием в канале сложного нетривиального трехмерного профиля скорости, проекции которого в сечениях по лучам и между лучами приведены на рис.4.



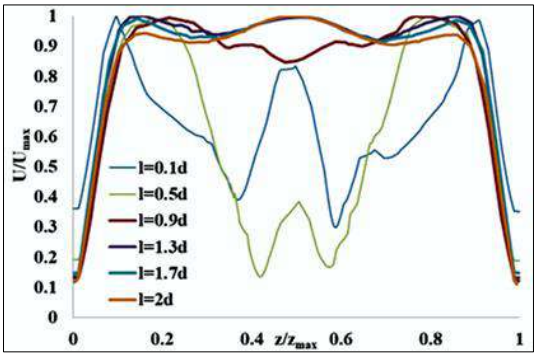
(а) (б)

Рис. 3. Вектора скорости в продольных сечениях канала массоподвода звездообразной формы по лучам (а) и между лучами (б)
Fig. 3. Velocity vectors in longitudinal sections of the star-shaped channel along the beams (a) and between the beams (b)

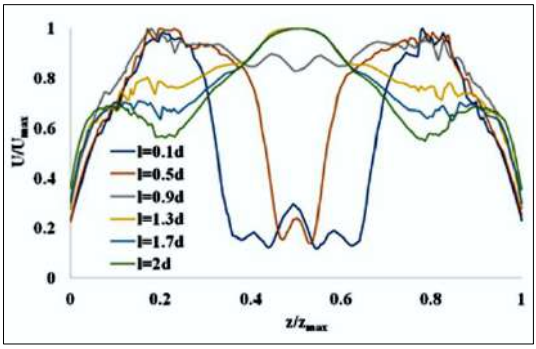
В цилиндрической части канала (рис. 3 (б)) наблюдается формирование вблизи поверхности вдува слоя смешения, что сопряжено с преобладанием потока газа, поступающего из лучей, над потоком газа, поступающим с цилиндрической (канальной) поверхности. Трансформация профиля скорости по длине канала представлена на рис. 4, так области смешения разнонаправленных потоков газа локализуются вблизи лучей канала. Это приводит к неравномерности профиля продольной компоненты скорости в поперечном сечении канала по лучам и между лучами звездообразного канала. Профиль продольной составляющей скорости в сечении между лучами звезды характеризуется большей равномерностью (рис.4 (б)) чем аналогичный профиль в сечении по лучам, что связано как с выбором плоскости разреза (по плоскости симметрии парного вихря, рис. 3 (б)), так и с характером течения газа в цилиндрической части канала (рис. 2).

При движении вниз по потоку форма профиля скорости в цилиндрической части канала массоподвода приближается к косинусоидальной форме, что согласуется (рис. 5) с экспериментальными данными [19, 20]. Профиль продольной компоненты скорости в сечении по лучам (рис. 4) имеет три локальных экстремума, характеризующих формирование

в канале трех зон локального увеличения скорости. Выявлено и показано, что две зоны повышения скорости формируются потоком, поступающим из лучей, центральная область повышения скорости формируется в основном (цилиндрическом) канале за счет смешения потоков (зоны I на рис. 3).



(а)

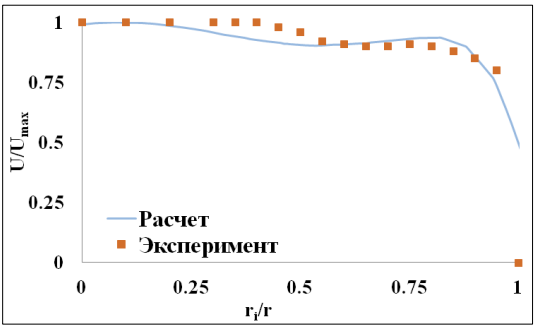


(б)

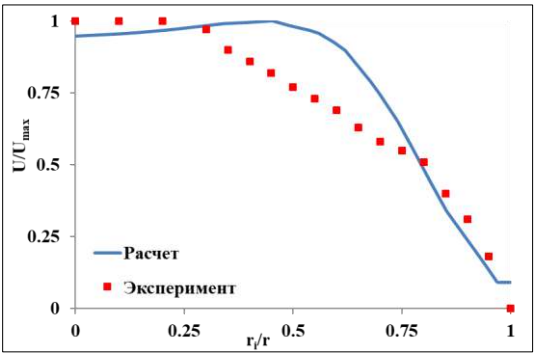
Рис. 4. Перестройка профиля скорости в поперечных сечениях звездообразного канала с массоподводом: (а) между лучами; (б) в сечении по лучам
Fig. 4. Velocity profile rearrangement in cross sections of star channel with mass flow: (a) between the beams; (b) in the cross section along the beams

Из рис. 4 видно, что при движении по потоку наблюдается выравнивание пространственного поля скорости с одной стороны и увеличение модуля скорости в плоскости лучей, с другой. В частности, выравнивание поля скорости сопровождается повышением его равномерности при отсутствии отрывных течений. В целом, анализ топологических особенностей потока газа в звездообразном канале позволяет однозначно определить связь между количеством, формой, расположением компенсаторов и пространственной локализацией парных вихревых структур в цилиндрическом проницаемом канале.

Представленные на рис. 5 расчетные профили продольной компоненты скорости удовлетворительно совпадают (качественно и количественно) с экспериментальными профилями [21, 22]. Необходимо отметить, что согласно [22] экспериментальная погрешность измерения скорости составляет 8%, а максимальное расхождение расчетных данных с результатами эксперимента составляет 12.5%.



(а)

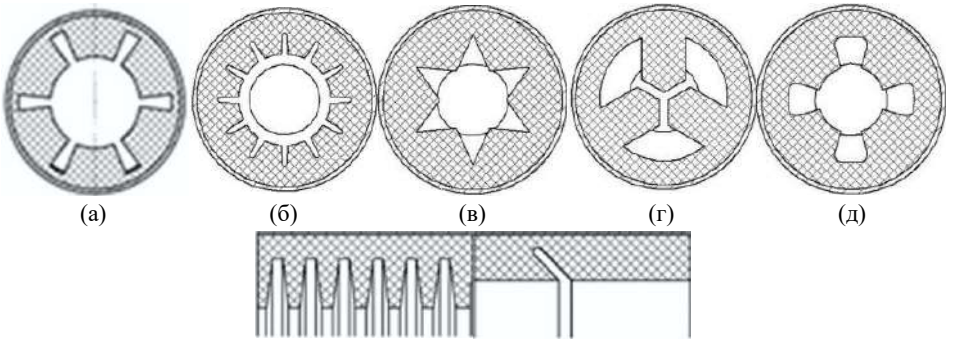


(б)

Рис. 5. Сравнение расчетного и экспериментального профилей продольной компоненты скорости в поперечных сечениях звездообразного канала со вдувом по длине канала (а) L/d = 1.5; (б) L/d=2.3

Fig. 5. Comparison of the calculated and experimental profiles of the longitudinal velocity component in the cross sections of a star-shaped channel with a blow along the length of the channel (a) L/d = 1.5; (b) L/d = 2.3

Рассмотрим особенности стационарного течения газа в каналах с массоподводом различных конфигураций (рис. 6).



(а) (б) (в) (г) (д)

(е) (ж)

Рис. 6. Форма поперечного сечения канала массоподвода: шестилучевой канал (а), двенадцатилучевой канал (б), шестилучевой звездчатый канал (в), трехлопастной канал (г), четырехлопастной канал (д), пилообразный канал (е) и канал с проточкой (ж)

Fig. 6. Cross-sectional shape of the mass channel: six-beam channel (a), twelve-beam channel (b), six-beam star channel (c), three-blade channel (d), four-blade channel (e), saw-tooth channel (f) and channel with a slot (g)

Выявлено, что при одинаковых расходах газа с поверхностей массоприхода в каналах исследуемых конфигураций реализуются локальные особенности структуры потока,

характерные строго для данного типа поперечного сечения канала с проницаемыми стенками. Так, для каналов с различными видами радиальных отводов/прорезей (рис. 7, а-в) в цилиндрической части канала фиксируется зона смешения, размеры которой определяются как количеством прорезей, так и газодинамическими характеристиками потока. Для лепестковых форм каналов массоподвода (рис. 7 (в-г)) характерно образование сложных пространственных конфигураций зон смешения в лепестковых трактах канала. Пилообразный (рис.7, е) канал, как и канал с кольцевой проточкой (рис. 7 (ж)) характеризуются сложной пространственной структурой течения в области резкого изменения геометрии (в области проточек/пропилов). Структура потока в поперечном сечении на выходе из канала массоподвода, для рассматриваемых конфигураций трактов, представлена на рис. 8.

Выявлено, что при малом числе отводов в областях взаимодействия канального и щелевых потоков образуются парные вихревые структуры, прослеживающиеся вдоль всего канала до выходного сечения. Увеличение числа отводов\пропилов приводит к изменению структуры потока в канале, а именно к отсутствию парных вихревых структур.

Переход от трапециевидной формы пропилов к треугольной (рис.7, а, в) приводит к изменению профиля продольной составляющей скорости на выходе из канала как в сечениях по лучам, так и между лучами (рис. 8, кривые 1 и 3). Анализ полученных профилей скорости позволяет сделать вывод об увеличении зоны смешения. В сечении по лучам в канальной части наблюдается трансформация профиля скорости от косинусоидального к параболическому (рис. 8, кривые 1 и 3).

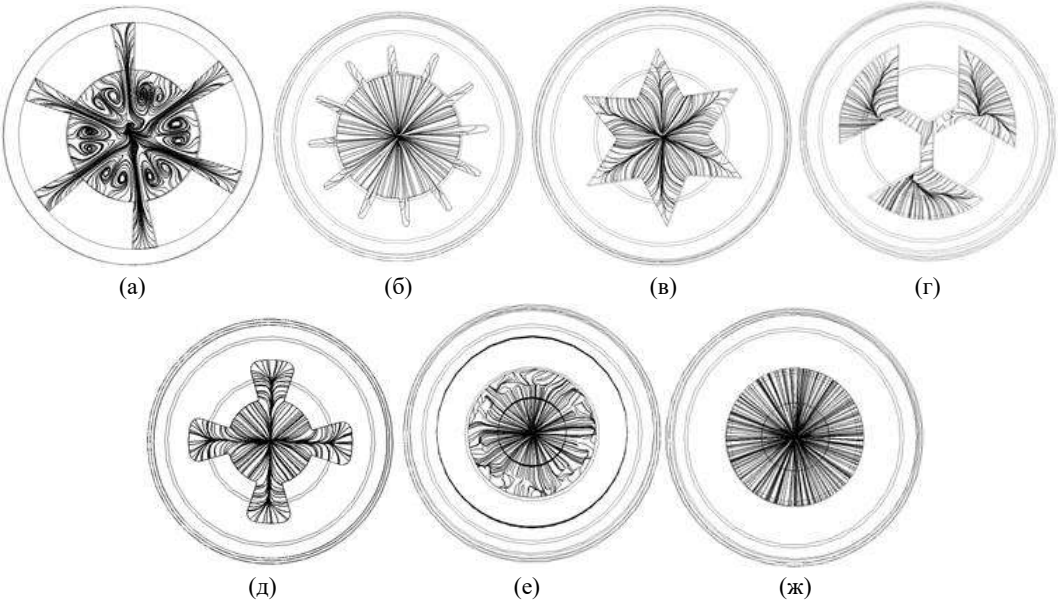


Рис. 7. Линии тока в шестилучевом канале (а), двенадцатилучевом канале (б), шестилучевом звездчатом канале (в), трехлопастном канале (г), четырехлопастном канале (д), пилообразном канале (е) и в канале с проточкой (ж)

Fig.7. Stream lines in the six-beam channel (a), twelve-beam channel (b), six-beam star channel (c), three-bladed channel (d), four-bladed channel (e), sawtooth channel (f) and channel with a slot (g)

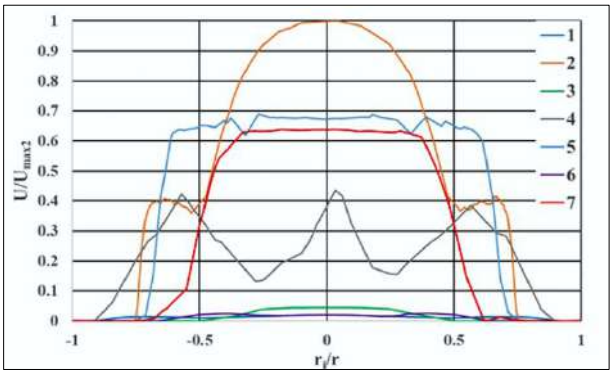


Рис. 8. Профиль продольной компоненты скорости на выходе из канала массоподвода где: 1 - шестилучевой канал; 2 - двенадцатилучевой канал; 3 - шестилучевой звездчатый канал; 4 - трехлопастной канал; 5 - четырехлопастной канал; 6 - пилообразный канал; 7 - канал с проточкой
Fig.8. Profile of the longitudinal velocity component at the outlet of the mass feed channel where: 1 - six-beam channel; 2 - twelve-beam channel; 3 - six-beam star channel; 4 - three-blade channel; 5 - four-blade channel; 6 - sawtooth channel; 7 - channel with a slot

Необходимо отметить, что при одинаковом массовом расходе газа с поверхности массоподвода переход от трапециевидной формы компенсаторов к треугольной приводит (рис. 8, кривые 1,3) к увеличению локальных максимальных значений продольной составляющей скорости на 54%. При этом, максимальные значения модуля скорости на выходе из канала достигаются в проницаемом канале цилиндрическо-щелевой формы с двенадцатью лучами (рис. 8, кривая 2). Использование четырехлепесткового канала (рис. 8, кривая 5) и канала с проточкой (рис. 8, кривая 7) приводит к формированию более равномерного и наполненного профиля скорости в выходном сечении канала, максимальное значение продольной составляющей скорости составляет 67% от аналогичного значения для двенадцатищелевого канала (кривая 2, рис. 8).

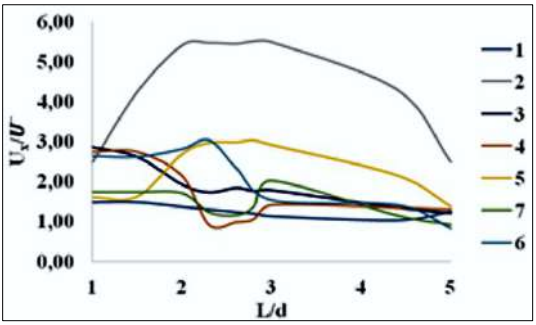


Рис. 9. Распределение коэффициента неравномерности по длине канала, где: где: 1 - шестилучевой канал; 2 - двенадцатилучевой канал; 3 - шестилучевой звездчатый канал; 4 - трехлопастной канал; 5 - четырехлопастной канал; 6 - пилообразный канал; 7 - канал с проточкой.
Fig.9. Distribution of the nonuniformity coefficient along the length of the channel, where: where: 1 - six-beam channel; 2 - twelve-beam channel; 3 - six-beam star channel; 4 - three-bladed channel; 5 - four-bladed channel; 6 - sawtooth channel; 7 - channel with a slot.

Влияние формы канала на изменение профиля скорости по длине канала удобно оценивать в рамках гидравлического подхода с использованием коэффициента неравномерности скорости $u_x/\bar{u}$ (рис. 9). Выявлено, что максимальная неравномерность профиля скорости в выходном сечении канала ($u_x/\bar{u} = 2.8$) характерна для двенадцатищелевого канала (рис. 9, кривая 2), а наименьшую неравномерность профиля скорости в выходном сечении канала

($u_x/\bar{u} \approx 1$) можно обеспечить за счет применения трехлепесткового канала и канала с кольцевой проточкой (рис. 9, кривые 3, 7).

4. Заключение

В работе исследованы вопросы влияния формы поперечного сечения канала со вдувом на реализуемые в канале профили продольной составляющей скорости. Выявлено, что коэффициент неравномерности скорости в выходном сечении канала имеет максимальные значения для двенадцатилучевого канала, а минимальные значения реализуются в каналах с трехлепестковой формой поперечного сечения и в канале с кольцевой проточкой. Учет неравномерности распределения продольной составляющей скорости на выходе из каналов необходим для последующего расчета процессов теплообмена в элементы конструкции энергетических установок с такими каналами. Кроме того, показаны особенности структуры потока в каналах с различной формой поперечного сечения, в частности, отмечены вихревые структуры вблизи поверхности вдува и зоны локального смешения потоков в каналах с радиальными проточками. Следует заметить, что выявленные вихревые структуры переносятся вниз по потоку и взаимодействуют с поверхностями элементов конструкции.

Список литературы / References

[1] Липанов А.М., Бобрышев В.П. и др. Численный эксперимент в теории РДТТ. Екатеринбург, УИФ Наука, 1994 г., 301 стр. / Lipanov A.M., Bobryshev V.P. et al. Numerical Experiment in SFRE theory. Ekaterinburg, UIF Nauka, 1994, 301 p. (in Russian).

[2] Алиев А.В., Мищенкова О.В. Математическое моделирование в технике. Ижевский институт компьютерных исследований, 2012 г., 476 стр. / Aliev A.V., Mischenkova O.V. Mathematical Modeling in Engineering. Izhevsk Institute of Computer Research, 2012, 476 p. (in Russian).

[3] Ахмадеев В.Ф., Сидоров А.Ф. и др. О трех методах расчета дозвуковых течений в осесимметричных каналах сложной формы. Моделирование в механике. Новосибирск, ИТПМ, том 4 (21), вып. 4, 1990 г., стр. 15-25 / Akhmadeev V. F., Sidorov A.F. et al. On three methods of calculation of subsonic flows in axisymmetric channels of complex shape. Modelling in Mechanics. Novosibirsk, ITAM, vol. 4 (21), 1990, pp. 15-25 (in Russian)

[4] Волков К.Н. Нестационарное турбулентное течение газовзвеси в канале при наличии вдува в условиях вынужденных колебаний давления. Прикладная механика и техническая физика, том 54, вып. 2, 2013 г., стр. 65-80 / Volkov K.N. Unsteady turbulent flow of gas suspension in the channel in the presence of blowing under forced pressure oscillations. Applied Mechanics and Technical Physics, vol. 54, issue 2, 2013, pp. 65-80 (in Russian).

[5] Kong F., Schetz Y.A. Turbulent boundary layer over solid and porous surface with small roughness. In Papers of the 19th Aerospace Sciences Meeting (1981), published online 2012. DOI: 10.2514/6.1981-418.

[6] Волков К.Н., Емельянов В.Н. Газовые течения с массоподводом в каналах и трактах энергоустановок. М., Физматлит, 2011 г., 464 стр. / Volkov K.N., Emelyanov V.N. Gas flows with mass transfer in channels and paths of power plants. Moscow, Fizmatlit, 2011, 464 p. (in Russian).

[7] Шумихин А.А., Королева М.Р. и др. Использование схемы WENO для моделирования турбулентного течения в канале с обратным уступом. Вестник Удмуртского университета. Математика. Механика. Компьютерные науки, том 27, вып. 3, 2017 г., стр. 460-469 / A.A. Shumikhin, M.R. Koroleva et al. Application of WENO scheme for simulation of turbulent flow in a channel with backward-facing step. The Bulletin of Udmurt University. Mathematics. Mechanics. Computer Science, vol. 27, issue 3, 2017, pp. 460-469 (in Russian).

[8] Mansour N.N., Kim J., Moin P. Reinolds-stress and dissipation rate budgets in a turbulent channel flow. Journal of Fluid Mechanics, vol. 194, 1988, pp. 15-44.

[9] Волков К.Н., Емельянов В.Н. Математические модели трехмерных турбулентных течений в каналах со вдувом. Математическое моделирование, том 16, вып. 10, 2004 г., стр. 41-63 / K.N. Volkov, V.N. Emelyanov. Mathematical models of three-dimensional turbulent flows in the ducts with fluid injection. Mathematical Modeling, vol. 16, issue 10, 2004, pp. 41-63 (in Russian).

[10] Benderskiy B.Y., Chernova A.A. Features of heat transfer in a pre-nozzle volume of a solid-propellant rocket motor with charges of complex shapes. Thermophysics and Aeromechanics, vol. 25, issue 2, 2018, pp. 265-272.
[11] Koroleva M.R., Mishchenkova O.V. et al. A Theoretical research of the internal gas dynamics processes of measurements of hot air curtain with cross-flow fan. MM Science Journal, June, 2020, pp. 3966-3972.
[12] Raeder T., Tenenev V., Chernova A., Koroleva M. Multilevel simulation of direct operated safety valve. In Proc. of the Ivannikov Ispras Open Conference (ISPRAS), 2018, pp. 109-115.
[13] Редер Т., Тененев В.А. и др. Численное моделирование газодинамики предохранительного клапана. Интеллектуальные системы в производстве, том 15, вып. 4, 2017 г., с. 4-11 / Raeder T., Tenenev V.A. et al. Numerical modeling of the gas dynamics of the safety valve. Intelligent Systems in Manufacturing, vol. 15, issue 4, 2017, pp. 4-11 (in Russian).
[14] Raeder T., Mishchenkova O.V. et al. Nonlinear processes in safety systems for substances with parameters close to a critical state. Russian Journal of Nonlinear Dynamics, vol. 17, issue 1, 2021, pp. 119-138.
[15] Chernova A.A. Validation of rans turbulence models for the conjugate heat exchange problem. Russian Journal of Nonlinear Dynamics, vol. 18, issue 1, 2022, pp. 61-82.
[16] Menter F.R., Kuntz M., Langtry R. Ten years of industrial experience with the SST turbulence model. Proc. of the Fourth International Symposium on Turbulence, Heat and Mass Transfer, 2003, pp. 625-632.
[17] Menter F.R. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. AIAA Journal, vol. 32, issue 8, 1994, pp. 1598-1605.
[18] Isaev S., Popov I. et al. Abnormal enhancement of separated turbulent air flow and heat transfer in inclined single-row oval-trench dimples at the narrow channel wall. Acta Astronautica, vol. 163, 2019, pp. 202-207.
[19] Benderskiy B., Chernova A., Frankovský P. Numerical simulation of intrachamber processes in the power plant. Applied Sciences, vol. 11, issue 11, 2021, article no. 4990.
[20] Benderskiy B.Y., Chernova A.A. Formation of vortex structures in channels with mass injection and their interaction with surfaces in solid-fuel rocket engines. Thermophysics and Aeromechanics, vol. 22, issue 2, 2015, pp. 185-190.
[21] Савельев С.К., Емельянов В.Н., Бендерский Б.Я. Экспериментальные методы исследования газодинамики РДТТ. СПб., Недра, 2007 г., 267 стр. / Savelyev S.K., Emelyanov V.N., Bendersky B.Ya. Experimental methods for studying the gas dynamics of solid propellant rocket engines. SPb, Nedra, 2007, 267p. (in Russian).
[22] Benderskii B. Ya., Tenenev V.A. Experimental and numerical investigation of flows in complex shaped axisymmetric channels with mass injection. Fluid dynamics, vol. 36, issue 2, 2001, pp. 336-340.

Информация об авторах / Information about authors

Борис Яковлевич БЕНДЕРСКИЙ – доктор технических наук, профессор кафедры Тепловые двигатели и установки. Сфера научных интересов: механика, процессы сопряженного теплообмена, двигатели летательных аппаратов, двигатели внутреннего сгорания, промышленный дизайн.

Boris Yakovlevich BENDERSKY – Doctor of Technical Sciences, Professor, Department of Thermal Engines and Installations. Research interests: mechanics, coupled heat exchange processes, aircraft engines, internal combustion engines, industrial design.

Алена Алексеевна ЧЕРНОВА – кандидат технических наук, доцент кафедры Тепловые двигатели и установки. Сфера научных интересов: газодинамика, гидродинамика, конвективный теплообмен, теплопередача, математическое моделирование.

Alena Alekseevna CHERNOVA – Candidate of Technical Sciences, Associate Professor, Department of Thermal Engines and Installations. Research interests: gas dynamics, hydrodynamics, convective heat exchange, heat transfer, mathematical modeling.

DOI: 10.15514/ISPRAS-2022-34(5)-14

Моделирование сопряженного теплообмена в микроканалах в среде OpenFOAM

Е.С. Байметова, ORCID: 0000-0002-4534-0936 <baimetova.e.s@gmail.com>
М.Е. Хвалько, ORCID: 0000-0001-8524-4749 <izhgazproekt@mail.ru>
А.Ю. Армянин, ORCID: 0000-0001-7541-7239 <armynin-izhevsk@mail.ru>
Ижевский государственный технический университет имени М.Т. Калашникова,
426069, Россия, г. Ижевск, ул. Студенческая, д. 7

Аннотация. В работе с использованием инструментов пакета OpenFOAM проведено численное моделирование вынужденного конвективного теплообмена в кремниевом микроканальном радиаторе. В качестве теплоносителя использовалась однофазная жидкость – вода. Модель микроканального радиатора представлена в виде кремниевой подложки длиной 10 мм, с прямоугольными микроканалами шириной 57 мкм и глубиной 180 мкм. расположенными по всей длине радиатора. Выполнен сравнительный анализ в виде кроссплатформенной верификации полученных численных результатов с данными сторонних авторов. Анализ полученных данных показал хорошую сходимость результатов исследования и возможность применения пакета OpenFOAM в качестве расчетной среды для численного моделирования физических процессов, протекающих в канальных радиаторах.

Ключевые слова: численное моделирование; микроканал; теплообмен; сравнительный анализ

Для цитирования: Байметова Е.С., Хвалько М.Е., Армянин А.Ю. Моделирование сопряженного теплообмена в микроканалах в среде OpenFOAM. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 205-214. DOI: 10.15514/ISPRAS-2022-34(5)-14

Modeling of coupled heat transfer in microchannels in OpenFOAM

E.S. Baymetova, ORCID: 0000-0002-4534-0936 <baimetova.e.s@gmail.com>
M.E. Hval'ko, ORCID: 0000-0001-8524-4749 <izhgazproekt@mail.ru>
A.Yu. Armyanin, ORCID: 0000-0001-7541-7239 <armynin-izhevsk@mail.ru>
Kalashnikov Izhevsk State Technical University
7, Studencheskaya st., Izhevsk, 426069, Russia

Abstract. In this paper, a numerical simulation of forced convective heat transfer in a silicon microchannel heat sink has been performed using the OpenFOAM tools. A single-phase fluid - water - was used as a heat transfer medium. The model of microchannel heat sink is represented as silicon substrate with length 10 mm, with rectangular microchannels 57 microns wide and 180 microns deep located along the full length of the heat sink. A comparative analysis in the form of cross-platform verification of the numerical results obtained with data from third-party authors was performed. The analysis of the obtained data has shown a good convergence of the study results and the possibility of using the OpenFOAM package as a computational environment for the numerical simulation of the physical processes occurring in channel radiators.

Keywords: numerical simulation; microchannel; heat exchange; comparative analysis.

For citation: Baymetova E.S., Hval'ko M.E., Armyanin A.Yu. Modeling of coupled heat transfer in microchannels in OpenFOAM. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 205-214 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-14

1. Введение

В настоящее время применение микромасштабных охлаждающих устройств является очень актуальной задачей теплоотвода и поддержания оптимальной температуры в микромеханических системах, для охлаждения электроники, в электрических цепях и других устройствах, которые подвергаются высоким тепловым нагрузкам. Микроканальные радиаторы позволяют обеспечивать высокие показатели коэффициента теплопередачи при течении жидкости в небольших объемах. Отличительная особенность микроканальных радиаторов заключается в том, что хладагент протекает через большое количество микроканалов, которые зачастую бывают связаны между собой сложным образом, что не позволяет разбивать теплообменник на отдельные составляющие и расчет приходится проводить целиком, что ведет за собой большую трату вычислительных ресурсов.

В работе [1] для решения задачи создания эффективного численного алгоритма для моделирования сопряженного теплообмена в микротеплообменниках предлагается использовать гибридный алгоритм, основанный на сочетании методов ТГЦ и пространственных методов вычислительной гидродинамики (CFD). В большинстве случаев решение гибридных задач осуществляется путем разделения пространственной и сетевой частей на разные расчетные модули и последующей их сшивки. Такой подход может быть неустойчивым и привести к значительному увеличению времени расчета. В случае с сопряженным теплообменом (твердое тело-жидкость) реализация гибридного подхода тем более затруднительна, так как сетевые элементы непосредственно включены в пространственную область. Авторами статьи [1] предложен метод, в котором подобная гибридная задача решается на основе общего уравнения на поправку давления. В данной статье представлены результаты тестирования методики на задачах гидродинамики и теплообмена в микроканалах.

В работе [2] рассматривается процесс прогрева стенки микроканала, для решения управляющих уравнений авторами был разработан конечно-разностный численный код с трехдиагональным матричным алгоритмом (TDMA), на основе которого проводился верификационный анализ при помощи решения задачи в среде OpenFOAM с использованием оригинальной схемы и применением исходных теплофизических характеристик исследуемых сред.

Численным расчетам в области микроканального моделирования посвящен ряд работ. В [3] рассматриваются процессы, которые необходимо учитывать при уменьшении диаметра каналов, включая проблемы с распределением потока в одиночном, параллельном и разделенном потоках, нестабильность потока в параллельных каналах, процессы зарождения и эффекты проводимости стенки.

В работе [4] коллектив авторов решал сопряженную задачу теплопередачи для микроканальных радиаторов, с целью получить подробные пространственные распределения температуры поперечного сечения радиатора по длине каналов в твердой подложке и жидкости, приведен расчетный алгоритм для выбора размеров теплообменников.

В [5] показан эксперимент по измерению коэффициента трения ламинарного потока деионизированной воды в гладких кремниевых микроканалах трапециевидного сечения с гидравлическими диаметрами в диапазоне 25,9–291,0 мкм. Доказано, что на постоянную трения этих микроканалов большое влияние оказывает соотношение сторон поперечного сечения микроканала. Установлено, что экспериментальные данные хорошо согласуются с существующим аналитическим решением для несжимаемого, полностью развитого ламинарного течения при граничном условии без скольжения. Подтверждено, что уравнения Навье–Стокса по-прежнему справедливы для ламинарного течения деионизированной воды в гладких кремниевых микроканалах с гидравлическим диаметром 25,9 мкм.

Применение методов численного моделирования для исследования и верификации физических процессов и результатов вычислений в технических устройствах является

актуальной задачей. Существует ряд пакетов прикладных программ для решения задач МСС, такие как SolidWorks, Ansys, FlowVision, OpenFOAM и др. Из представленного списка пакет OpenFOAM является полноценным, завершенным, свободно распространяемым программным продуктом с открытым исходным кодом и набором библиотек, на основе которых можно разработать полноценные численные алгоритмы для решения задач МСС.

Зачастую проведение натурных экспериментов невозможно или экономически не целесообразно, в связи с этим применяются численные эксперименты, в результате которых можно получить достаточно физичную картину процессов, протекающих в исследуемой задаче. Численный эксперимент должен быть подкреплен верификацией с натурным экспериментом или кроссверифицирован.

Целью работы является проведение кроссплатформенной верификации результатов численного эксперимента, проведенного в OpenFOAM, с результатами исследований представленных в работе [2].

2. Метод решения

Для решения задачи сопряженного теплообмена используется решатель *chtMultiRegionSimpleFoam*, который является объединением программ расчета тепломассопереноса в текучих средах *buoyantSimpleFoam* и процессов теплопроводности в твердом теле на основе уравнения Лапласа *laplacianFoam*. Основные уравнения сохранения, решаемые с помощью метода контрольных объемов можно записать в следующей форме [6-8]:

$$\nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{1}$$

$$\nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \rho \boldsymbol{g} + \nabla \cdot \left(2\mu_{eff}\boldsymbol{D}(\boldsymbol{u})\right) - \nabla\left(\frac{2}{3}\mu_{eff}(\nabla \cdot \boldsymbol{u})\right), \tag{2}$$

где ρ – плотность среды, $\boldsymbol{u}(u, v, w)$ – вектор скорости, p – статическое давление, $\boldsymbol{g}$ – ускорение свободного падения, μ_{eff} – эффективная вязкость, представляющая собой сумму молекулярной вязкости (динамического коэффициента вязкости жидкости) μ и турбулентной вязкости $\rho\nu_t$, $\boldsymbol{D}(\boldsymbol{u})$ – тензор скорости деформации, определяющийся как $\boldsymbol{D}(\boldsymbol{u}) = \frac{1}{2}(\nabla u + (\nabla u)^T)$.

Уравнение энергии может быть выражено с использованием внутренней энергии e следующим образом:

$$\nabla \cdot (\rho \boldsymbol{u} e) + \nabla \cdot (\rho \boldsymbol{u} K) + \nabla \cdot (p\boldsymbol{u}) = \nabla \cdot \left(\alpha_{eff}\nabla e\right) + \rho \boldsymbol{u} \cdot \boldsymbol{g}, \tag{3}$$

где $K \equiv |\boldsymbol{u}|^2/2$ – кинетическая энергия на единицу массы, α_{eff} – эффективная температуропроводность: $\alpha_{eff} = \rho\nu_t/Pr_t + \mu/Pr = \rho\nu_t/Pr_t + k/c_p$, где k – теплопроводность, c_p – удельная теплоемкость, Pr– число Прандтля, Pr_t– турбулентное число Прандтля.

Кондуктивный теплообмен через твердое тело может быть определен с помощью решателя *laplacianFoam*, для которого основное уравнение записывается следующим образом:

$$\frac{\partial T}{dt} = \nabla \cdot (\alpha \nabla T), \tag{4}$$

где α– коэффициент термической диффузии $\alpha = k/(\rho c_p)$, ρ – плотность тела, c_p – удельная теплоемкость твердого тела. Когда решается стационарная задача, слагаемое в левой части уравнения (4), зависящее от времени не учитывается.

Алгоритм решения задачи в виде блок-схемы приведен на рис. 1. С помощью утилиты построения сетки *blockMesh* создается сетка для общего домена. Затем, проводится разделение областей, занятых жидкостью и твердым телом с использованием утилит *topoSet* и *splitMeshRegions*. На этом этапе проводится определение границ контакта жидкости и

твердого тела, создаются отдельные директории в проекте, позволяющие задавать разные физические свойства веществ, начальные и граничные условия.

Рис. 1. Ход решения задачи
Fig. 1. Work flow of task

3. Постановка задачи

Объектом исследования являются процессы сопряженного теплообмена в системе микроканалов кремниевого радиатора. По системе параллельных микроканалов, вытравленных в кремниевой пластине, которая примыкает к нагреваемому устройству, пропускается охлажденная жидкость (рис. 2а). Задача решается для единичного канала с условиями симметрии по боковым стенкам. Расчетная область с размерами приведена на рис. 2б. Геометрия радиатора, расположение и размеры каналов соответствуют работе [2].

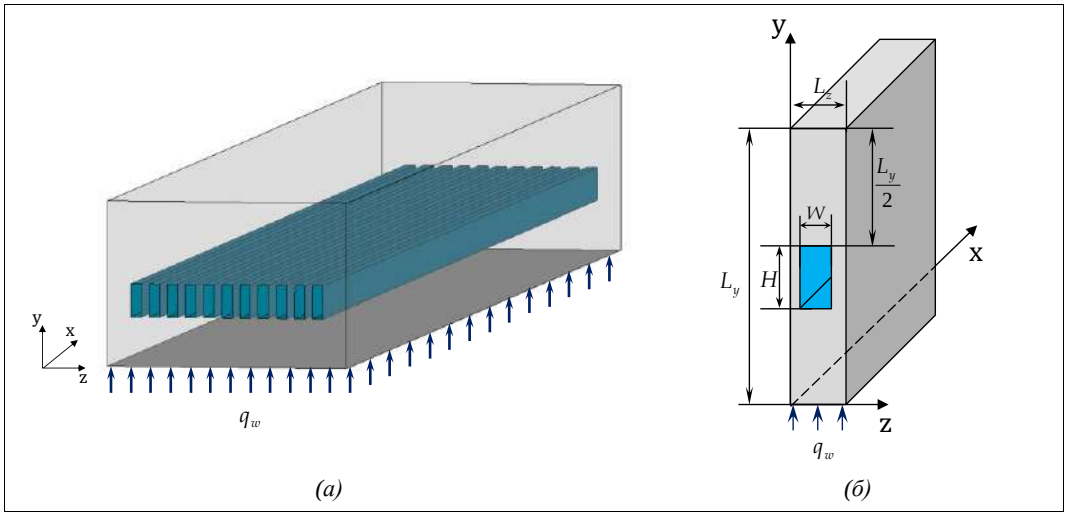


Рис. 2. Геометрическая постановка задачи:
(а) схема микроканального радиатора, (б) схема расчетной области
Fig. 2. Geometric problem description:
(a) microchannel radiator scheme, (b) calculation domain scheme

Геометрические размеры радиатора, представленного на рис. 2: длина L_x, высота L_y, ширина L_z равны 10 мм, 0.9 мм и 0.1 мм соответственно. Высота каналов охлаждения H равна 0.18 мм, ширина $W = 0.057$ мм, расстояние между каналами составляет $l = 0.043$ мм.

Корпус радиатора выполнен из кремния с плотностью 2330 кг/м^3, теплоемкостью 714 Дж/(кг*К), теплопроводностью 149 Вт/(м*К). В качестве охлаждающей среды принята вода с плотностью 1000 кг/м^3, теплоемкостью 4183 Дж/(кг*К), молекулярной вязкостью 1.004*10^{-3} м^2/с, число Прандтля в расчетах бралось равным 7.02.

Расчетная область (рис.3) включает границу подвода тепла (1), границы симметрии (2), адиабатические непроницаемые стенки (3), границы контакта жидкости и твердого тела (4), границы подвода (5) и отвода (6) жидкости.

Для твердого тела:

1: $Y = 0, q_w = 90$ Вт/см^2;

2: $Z = 0, Z = L_z, \frac{\partial f}{\partial n} = 0$;

3: Y= $L_y, X = 0, X = L_x$ (за исключением границ 5 и 6), $\frac{\partial T}{\partial n} = 0$.

Граница контакта жидкости с твердым телом:

4: $0 \le X \le L_x,\ Y = L_y/2 - H,\ Y = L_y/2,\ Z = l/2Z = l/2 + W,\ u = v = w = 0,\ -\lambda_w \frac{\partial T_w}{\partial \boldsymbol{n}} = -\lambda_f \frac{\partial T_f}{\partial \mathbf{n}}, T_w = T_f$, где индексы «$f$» и «$w$» относятся к жидкости и твердому телу, соответственно.

Для жидкости:

5: $X = 0, L_y/2 - H \le Y \le L_y/2, p = 50$ кПа, $T = 293\ K$,

6: $X = L_x, L_y/2 - H \le Y \le L_y/2, p = 0$ Па.

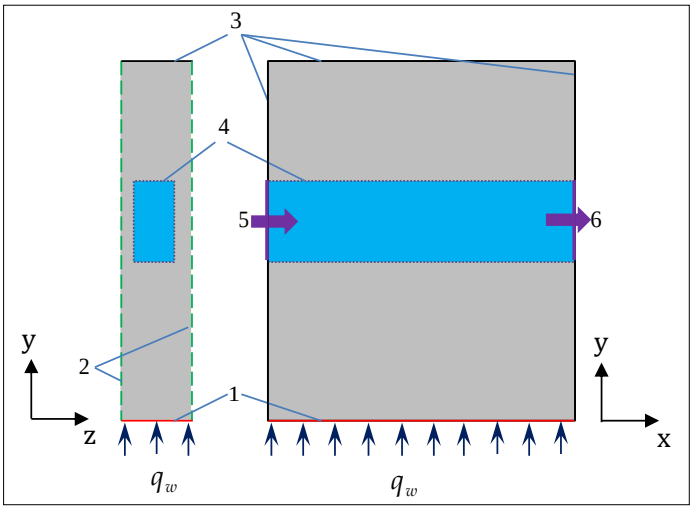


Рис. 3. Граничные условия
Fig. 3. Boundary conditions

Жидкость двигается по каналу под действием перепада давления 50 кПа. При таком перепаде согласно [2] поток остается ламинарным и характеризуется числом Рейнольдса равным 144, которое определяется по эквивалентному диаметру D и средней скорости жидкости. Для данного канала значение D, определяемое по формуле $D = \frac{4WH}{2(W+H)}$ равно 0.8658 мкм.

Ввиду простоты геометрии расчетной области и регионов, выделенных под жидкость и твердое тело, сетка состояла из множества параллелепипедов. На объем жидкости пришлось 90 тыс. ячеек, на твердое тело – 720 тыс. ячеек. Внутри жидкого региона ячейки были одинакового размера. Сетка твердого домена строилась так, чтобы на границах сопряжения сетки были полностью согласованными.

4. Анализ результатов численного моделирования

В результате численного моделирования получены распределения полей физических величин в рабочих объемах микроканалов радиатора. Так, на рис. 4 приведены линии равных скоростей в выходном поперечном сечении микроканала. В целом, наблюдается установившееся равномерное течение жидкости, подчиняющееся классическим представлениям о течении жидкости в прямолинейных каналах прямоугольной формы [9]. Видно хорошее качественное соответствие изотах, полученных с использованием матмодели (1)-(4) и средств OpenFOAM (черные линии на рис. 4) с результатами численного моделирования, представленными в работе [2] (цветные линии). Средняя скорость воды на выходе равняется 1.1026 м/с, а максимальная скорость составляет 1.97985 м/с, что также соответствует результатам работы [2]. Рассчитанное по данным значениям число Рейнольдса равно 95.548 и 171.415 для средней и максимальной скорости. Расхождение с экспериментальными и теоретическими значениями чисел Рейнольдса из работы [2] не превышает 0.5%.

Распределение линий равных температур в поперечном выходном сечении микроканала представлено на рис. 5. Видна некоторая асимметрия изотерм со смещением локального минимума вверх по поперечной координате, что обусловлено разностью температур между нижней и верхней стенками радиатора. При этом полученные с использованием сформированной математической модели (1)-(3) и средств OpenFOAM (черные линии на рис. 5) хорошо согласуются с результатами [2] в центральной зоне и на периферии вблизи верхней стенки канала. В работе [2] отмечено локальное повышение температуры жидкости в углах

канала, что также показано и на рис. 5. Однако, характер, а именно угол наклона, изотерм, полученных в результате используемого в настоящей работе подхода отличен от изотрем, представленных в работе [2]. В частности, наблюдается более интенсивный отвод тепла в стенки радиатора, что может быть связано с расхождением в задании физических свойств кремния. Максимальный прогрев воды в опорной работе составляет 32°C. В текущих расчетах максимальная средняя температура воды равна 31.08°C. В этом случае относительная погрешность составляет 2.8%.

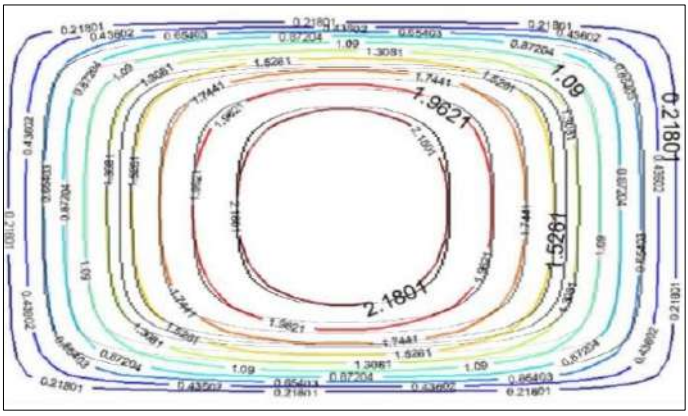


Рис. 4. Сопоставление распределения изотах на выходе из канала: черные линии – численный расчет в OpenFOAM; цветные – расчет [2]
Fig. 4. Comparison of the distribution of isotopes at the outlet of the channel: black lines – numerical calculation in OpenFOAM; color lines – results from work [2]

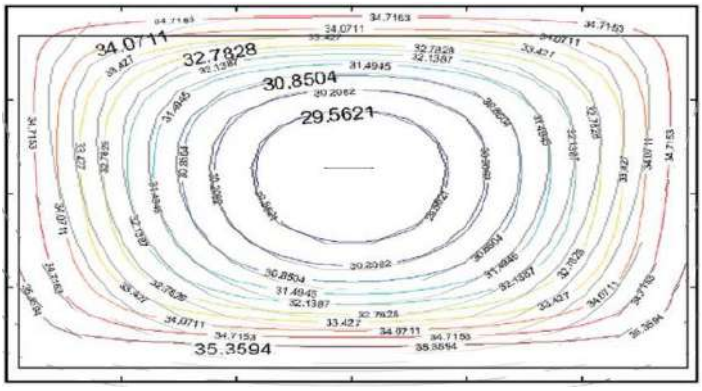


Рис. 5. Сопоставление распределения изотерм в канале (черные линии – численный расчет в OpenFOAM; цветные – расчет [2])
Fig. 5. Comparison of the distribution of isotherms in the channel (black lines – numerical calculation in OpenFOAM; color lines – calculation [2])

На рис. 6 приведено распределение температуры по продольным сечениям расчетной области. Рис. 6а показывает температурное поле на грани симметрии в плоскости $Z = 0$ или $Z = L_z$. Рис. 6б соответствует средней плоскости $Z = L_z/2$ и иллюстрирует распределение не только в твердой стенке, но и в жидкости. Охлаждение кремниевого радиатора происходит неравномерно. Наиболее интенсивно охлаждается первая треть объема, затем интенсивность снижается ввиду прогрева жидкости. Движение жидкости влияет на температурное поле радиатора, это видно по наклону линий изотерм в нижней и верхней части радиатора. Наиболее сильный градиент соответствует первой трети канала, что говорит о сильном отводе тепла от нижней нагреваемой части радиатора.

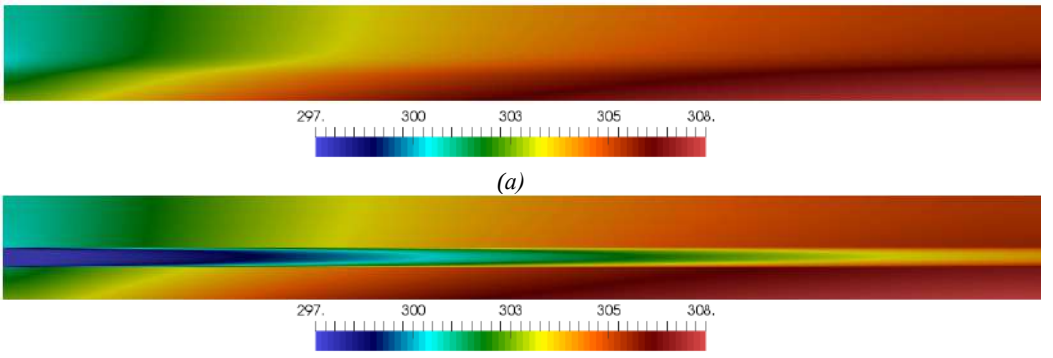
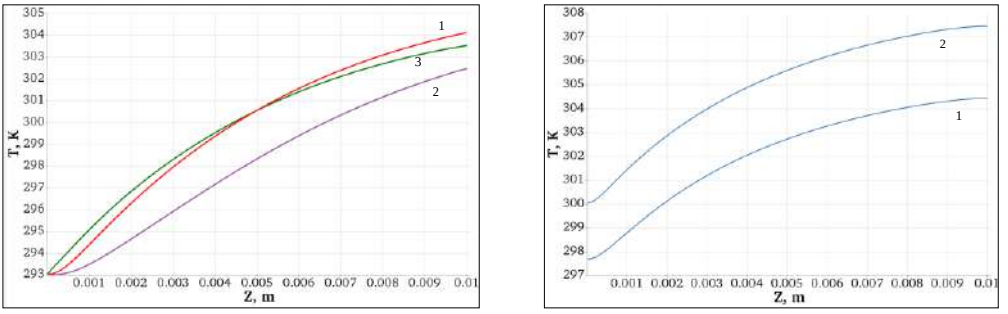


Рис. 6. Распределение температуры в продольных сечениях радиатора:
(а) плоскость симметрии, $Z = 0,\ Z = L_z$; (б) центральная плоскость, $Z = L_z/2$
Fig. 6. Temperature field in the radiator cross section:
(a) symmetry plane at $Z = 0, Z = L_z$; (b) central plane at $Z = L_z/2$

Изменение температуры жидкости и в стенках радиатора по каналу показано на графиках рис. 7. Три линии на рис. 7а показывают температуру воды вдоль нижней стенки, на середине канала и на верхней стенке канала. Вначале канала температура воды выше на верхней стенке, по всей видимости, сказывается нагрев радиатора между каналами. В середине канала в точке $Z = 0.005$ м верхняя и нижняя стенки имеют одинаковую температуру, далее нижняя стенка начинает прогреваться быстрее. Однако разница температур к концу канала достигает всего 1 К. Температура в ядре потока всегда ниже, чем температура жидкости на стенках, максимальная разница в центре канала составляет 3 К. На рис. 7б видно, что верхняя, относительно канала с водой, часть радиатора имеет более низкую температуру по сравнению с нижней частью. По длине канала эта разница несущественно, но увеличивается. В начале канала разница составляет 2 К, в конце – 3 К.



(а) *(б)*

Рис. 7. График изменения температуры вдоль радиатора для:
(а) воды:1 – нижняя стенка, 2 – середина канала, 3 – верхняя стенка,
(б) твердого тела: 1 – верхняя часть радиатора (над каналом), 2 – нижняя часть (под каналом)
Fig. 7. Temperature plots along radiator:
(a) for water in channel: 1 – bottom wall, 2 – middle channel, 3 – top wall,
(b) for solid: 1 – top radiator part, 2 – bottom radiator part

Температурное поле во входном и выходном сечениях радиатора приведено на рис.8. Температура стенки, к которой осуществляется подвод тепла равна 300 К во входном и 308 К в выходном сечении. Температура верхней стенки равна 297.5 К и 304.8 К соответственно. Таким образом во входном сечении разница температур на верхней и нижней стенках радиатора составляет 2.5 К, а в выходном сечении – 3.2 К. Средняя температура

прогреваемой стенки составляет 305.4 К, а верхней стенки радиатора – 302 К, т.е. температура в среднем понижается на 3.4 К. Температура воды в ядре потока меняется с 293 К до 302 К. Средний прогрев жидкости в микроканале составляет 11 К. Количество тепла, отводимое водой, составило 14 Вт/см^2.

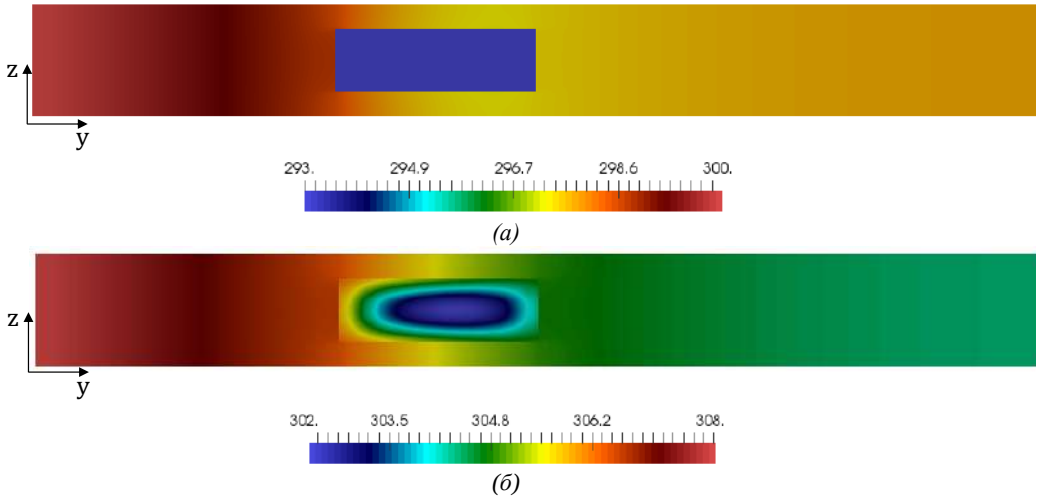


Рис. 8. Распределение температуры во входном (а) и выходном (б) сечении радиатора
Fig. 8. Temperature field in the inlet (a) and the outlet (b) section of the radiator

5. Заключение

В работе исследован процесс сопряженного теплообмена в микроканальном радиаторе. Выявлено, что реализуемые в микроканалах течения подчиняются законам континуальной физики сплошных сред и могут быть корректно смоделированы с применением предложенной математической модели с учетом вязких эффектов рабочей жидкости. Проведенная кроссплатформенная верификация показала хорошую сходимость результатов численного моделирования. Это подтверждает корректность предложенных подходов в целом и возможность применения пакета OpenFOAM, в частности, для математического моделирования процесса сопряженного теплообмена в микроканальных радиаторах.

В результате проведенных расчетов, была получена общая картина температурного режима в объеме кремниевого радиатора с микроканалами, по которым прокачивается охлаждающая жидкость. Получены локальные и интегральные характеристики работы радиатора.

Список литературы / References

[1] Филимонов С.А., Дектерев А.А. и др. Моделирование сопряженного теплообмена в системе микроканалов при помощи гибридного алгоритма. Сибирский журнал индустриальной математики, том 18, вып. 3, 2015 г., стр. 86-97 / Filimonov S.A., Dekterev A.A. et al. Simulation of conjugate heat transfer in a microchannel system by a hybrid algorithm. Journal of Applied and Industrial Mathematics, vol. 9, issue 4, 2015, pp. 469-479.

[2] Li J., Peterson G.P., Cheng P. Three-dimensional analysis of heat transfer in a micro-heat sink with single phase flow. International Journal of Heat and Mass Transfer, vol. 47, issues 19-20, 2004, pp. 4215-4231.

[3] Kendall G.E., Griffith P. et al. Small diameter effects on internal flow boiling. In Proc. of the 2001 ASME International Mechanical Engineering Congress and Exposition, 2001, pp. 1-17.

[4] Weisberg A., Bau H.H., Zemel J.N. Analysis of microchannels for integrated cooling. International Journal of Heat and Mass Transfer, vol. 35, issue 10, 1992, pp. 2465-2474.

[5] Wu H.Y., Cheng P. Friction factors in smooth trapezoidal silicon microchannels with different aspect ratios. International Journal of Heat and Mass Transfer, vol. 46, issue 14, 2003, pp. 2519-2525.

[6] Li Yu. Implementation of multiple time steps for the multi-physics solver based on chtMultiRegionFoam. In Proceedings of CFD with OpenSource Software, 2016, 50 p.
[7] Koroleva M.R., Mishchenkova O.V. et al. A Theoretical research of the internal gas dynamics processes of measurements of hot air curtain with cross-flow fan. MM Science Journal, June 2020, pp. 3966-3972.
[8] Байметова Е.С., Гиззатуллина А.Ф., Пушкарев Ф.Н. Решение задачи сопряженного теплообмена в оребренной трубке с использованием OpenFoam. Химическая физика и мезоскопия, том 23, вып. 2, 2021 г., стр. 154-164 / Baimetova E.S., Gizzatullina A.F., Pushkarev F.N. Solving the conjugate heat transfer problem in the ribbed tube with OpenFoam. Chemical physics and mesoscopy, vol. 23, issue 2, 2021, pp. 154-164 (in Russian).
[9] Лойцянский Л.Г. Механика жидкости и газа. М., Наука, 1970 г., 904 стр. / Loytsyansky L.G. Mechanics of liquid and gas. Moscow, Nauka, 1970, 904 p. (in Russian).

Информация об авторах/ Information about authors

Елена Сергеевна БАЙМЕТОВА – старший преподаватель кафедры «Тепловые двигатели и установки». Сфера научных интересов: вычислительная гидрогазодинамика и интенсификация процессов теплообмена.

Elena Sergeevna BAIMETOVA is a senior lecturer at the Department of Heat Engines and Plants. Research interests: computational fluid dynamics and intensification of heat exchange processes.

Михаил Евгеньевич ХВАЛЬКО – аспирант кафедры «Тепловые двигатели и установки». Сфера научных интересов: газодинамика, гидродинамика, конвективный теплообмен.

Mikhail Evgenevich HVAL'KO is a Postgraduate Student at the Department of Heat Engines and Plants. Research interests: gas dynamics, hydrodynamics, convective heat transfer.

Алексей Юрьевич АРМЯНИН – аспирант кафедры «Тепловые двигатели и установки». Сфера научных интересов: гидродинамика, конвективный теплообмен.

Aleksej Yurevich ARMYANIN is a Postgraduate Student at the Department of Heat Engines and Plants. Research interests: hydrodynamics, convective heat transfer.

DOI: 10.15514/ISPRAS-2022-34(5)-15

Особенности построения сетки для моделирования процесса обледенения треугольного крыла сложной формы

К.Б. Кошелев, ORCID: 0000-0002-7124-3945 <k.koshelev@ispras.ru>
А.В. Осипов, ORCID: 0000-0001-9223-4274 <a.osipov@ispras.ru>
С.В. Стрижак, ORCID: 0000-0001-5525-5180 <s.strijhak@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В статье рассматривается задача о моделировании обледенения треугольного крыла самолета-демонстратора X-59. Рассматривается три варианта сеток на 1.1, 3.8, 9.6 млн. ячеек. Обледенение крыла моделируется с помощью решателя iceFoam, разработанного в рамках пакета OpenFOAM. Для решения задачи используется две сетки: первая во внешней области, вторая - для жидкой пленки у твердого тела. Для построения сетки газовой фазы используются утилиты blockMesh и snappyHexMesh в составе пакета OpenFOAM. Качество сетки газовой фазы, определяемое стандартной утилитой checkMesh, соответствует всем проверяемым требованиям. Однако, в ходе автоматического построения сетки жидкой пленки могут образовываться ячейки с неудовлетворительными параметрами, к которым, например, относится требование ограниченной неортогональности граней. В связи с этим обсуждается новый алгоритм исключения некачественных расчетных ячеек. Расчеты были проведены для моделей крыла самолета-демонстратора X-59 в масштабе 1:25 и 1:1 для случая рыхлого льда. Для разномасштабных моделей обеспечивалось неизменность чисел Рейнольдса и Маха. В тоже время неизменными параметрами в размерном виде были водность воздуха и медианный диаметр капель воды. Получены картины образования льда на верхней и нижней части крыла. Показано, что области обледенения разномасштабных моделей крыла могут существенно различаться даже при совпадении безразмерных комплексов подобия для газовой фазы. Сделан вывод о том, что многие экспериментальные и расчетные результаты по обледенению профилей небольшого размера трудно перенести на полномасштабные профили. Вычисления выполнялись на кластере ИСП РАН с использованием 48 или 96 ядер процессоров.

Ключевые слова: обледенение; крыло; моделирование; сетка; ячейка; решатель; алгоритм

Для цитирования: Кошелев К.Б., Осипов А.В., Стрижак С.В. Особенности построения сетки для моделирования процесса обледенения треугольного крыла сложной формы крыла. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 215-226. DOI: 10.15514/ISPRAS-2022-34(5)-15

Благодарности: Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-29-13016.

Features of creating a mesh for modeling the icing process of a delta-wing with a complex shape

K.B. Koshelev, ORCID: 0000-0002-7124-3945 <k.koshelev@ispras.ru>
A.V. Osipov, ORCID: 0000-0001-9223-4274 <a.osipov@ispras.ru>
S.V. Strijhak, ORCID: 0000-0001-5525-5180 <s.strijhak@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. The article deals with the problem of modeling the icing of a delta-wing of the X-59 demonstrator aircraft. Three variants of grids for 1.1, 3.8, 9.6 million cells are considered. Wing icing is modeled using the iceFoam solver developed as part of the OpenFOAM package. To solve the problem, two grids are used: the first in the outer region, the second for a liquid film near a solid. blockMesh and snappyHexMesh utilities as part of the OpenFOAM package are used to build the gas phase grid. The quality of the gas phase grid, determined by the standard checkMesh utility, meets all the requirements being checked. However, during the automatic construction of the liquid film mesh, cells with unsatisfactory parameters may be formed, which, for example, include the requirement of limited non-orthogonality of the faces. In this regard, a new algorithm for excluding low-quality calculation cells is being discussed. The simulations were carried out for the X-59 demonstrator wing models on a scale of 1:25 and 1:1 for the case of loose ice. For different-scale models, the invariance of the Reynolds and Mach numbers was ensured. At the same time, the constant parameters in the dimensional form were the water content of the air and the median diameter of water droplets. Patterns of ice formation on the upper and lower parts of the wing were obtained. It is shown that the icing regions of different-scale wing models can differ significantly even when the dimensionless similarity complexes for the gas phase coincide. It is concluded that many experimental and calculated results on the icing of small profiles are difficult to transfer to full-scale profiles. The simulation was performed on the ISP RAS cluster using 48 or 96 processor cores.

Keywords: icing; wing; modelling; mesh; cell; solver; algorithm

For citation: Koshelev K.B., Osipov A.V., Strijhak S.V. Features of creating a mesh for modeling the icing process of a delta-wing with a complex shape. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 215-226 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-15

Acknowledgements. The reported study was funded by RFBR, project number № 19-29-13016.

1. Введение

В настоящее время проектируются новые перспективные сверхзвуковые пассажирские самолеты. В связи с этим ведутся работы по проектированию и испытаниям самолета-демонстратора. Например, действующий проект самолета-демонстратора Lockheed Martin X-59 [1]. Данный самолет-демонстратор совершает посадку и взлет на определенных высотах на дозвуковых скоростях. В процессе движения самолета в облаках существует вероятность возникновения процесса обледенения, которое является опасным явлением, так как снижает аэродинамические характеристики самолета. Образовавшийся лед бывает разных типов и формы: рыхлый, прозрачный, барьерный, смешанный, рогообразный, шероховатый. Поэтому необходимо проводить научные исследования по изучению данного явления.

Одним из возможных подходов является моделирование нарастания льда с помощью физических и численных моделей. Для решения сформулированной краевой задачи необходимо построить расчетную область и выполнить генерацию сетки. В данной работе для нас было важно повысить качество поверхности цифровой модели самолета-демонстратора X-59 и этап построения поверхностной неструктурированной сетки в области передней кромки треугольного крыла с затуплением.

2. Цифровая модель крыла

В открытом доступе имеется 3D цифровая модель самолета-демонстратора X-59 в формате STEP [2]. В данной работе рассматривалось только треугольное крыло сложной формы (Рис. 1, 2). Цифровая модель самолета-демонстратора X-59 была обработана в пакете Salome и преобразована в формат STL. Было выполнено улучшение качества поверхности крыла с целью сокращения количества внутренних объектов. Для решения краевой задачи были определены габариты расчетной области в форме параллелепипеда. Крыло находилось в центре расчетной области. Треугольное крыло имело сложную форму с круткой и дополнительной поверхностью для расположения механизации крыла. Построение сетки выполнено с использованием утилит blockMesh и snappyHexMesh в составе пакета OpenFOAM. Рассматривалось дозвуковое обтекание крыла при заданных значениях скорости, температуры, диаметра капель, водности LWC, угла атаки 0°.

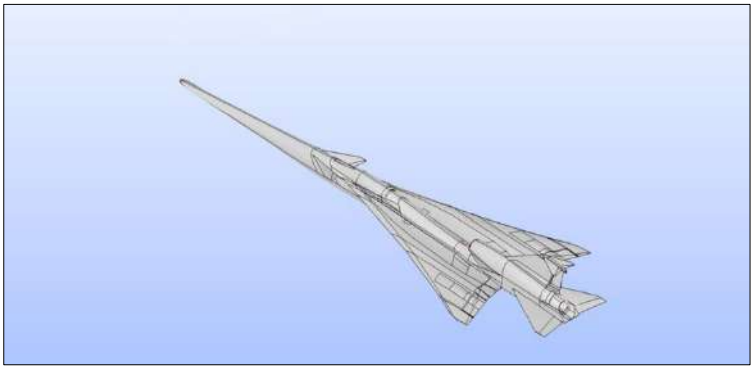


Рис. 1 Модель самолета X-59
Fig. 1 X-59 model

Крыло имело следующие параметры для масштаба 1:25:

- длина концевой хорды крыла $b_к = 0.03$ м.;
- длина бортовой хорды крыла $b_б = 0.398$ м.;
- угол стреловидности по передней кромки крыла составляет $\chi_{п.к} = 74^0$;
- удлинение крыла $\lambda=1.06$;
- геометрическая крутка крыла 7^0;
- относительная толщина $\overline{c} = 5.6$ %;
- сужение крыла $\eta=13.2$.

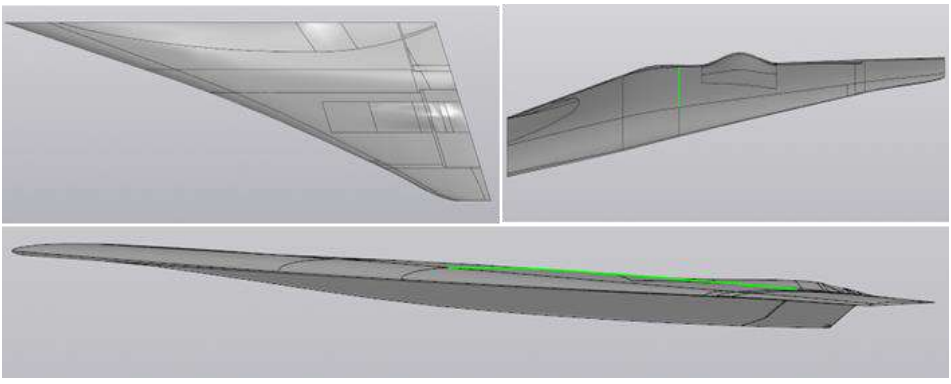


Рис.2 Модель крыла самолета X-59
Fig.2 Model of wing for X-59

С методикой расчета параметров для стреловидного крыла можно ознакомиться в монографии авторов из ЦАГИ им. Н.Е. Жуковского [3].

3. Математическая модель

Ранее в эксперименте исследовалось обледенение стреловидного крыла с несимметричным профилем GLC-305 в аэроклиматической трубе [4]. Для изучения процесса обледенения нами использовался решатель iceFoam в Эйлер-Лагранжевой постановке, который был разработан в ИСП РАН. В решателе реализована модель жидкой пленки по теории мелкой воды для описания термодинамической модели формирования льда, модуль для перестроения геометрии тела, модуль для работы с динамической сеткой, модель URANS и модель турбулентности k-ω SST с пристеночными функциями [5,6,7].

В ходе проведения исследования выяснилось необходимость в новом алгоритме выявления и устранения некачественных ячеек при построении расчетной сетки. Особенностью построения сетки для решателя iceFoam является двухэтапный процесс. Сначала с использованием какого-либо генератора сеток (например, утилиты snappyHexMesh) строится сетка для газовой фазы, а затем, с помощью утилиты extrudeToRegionMesh генерируется сетка для пленки в автоматическом режиме. Оказалось, что достаточно хорошее качество элементов сетки (при разрешении геометрии крыла сложной формы) для несущей фазы не гарантирует столь же хорошее качество сетки для пленки. Было выполнена реализация алгоритма выявления координат «плохих-некачественных» узлов и их визуализации в программе ParaView.

4. Постановка задачи

С целью первой оценки выбранного подхода рассматривалась модель треугольного крыла X-59 в масштабе 1:25. Также для нас было важно оценить доступные вычислительные ресурсы, поэтому было выполнено уменьшения количества расчетных ячеек. Было введено ограничение по количеству ячеек – 10 млн. ячеек.

Построение сетки осуществлялось встроенной утилитой snappyHexMesh входящей в состав пакета OpenFOAM v2012. Так как расчёты проводились на динамической сетке, то есть в процессе решения, сетка вблизи и на поверхности крыла перестраивалась, то была поставлена задача сгенерировать равномерную сетку с наименьшим числом ячеек, которые могли привести к снижению скорости счета или вообще к расхождению решения.

Геометрическая форма исследуемого тела имела достаточно сложную форму, имелось сужение крыла, крутка и выпуклости. Для достижения удовлетворительного разрешения геометрической формы крыла с учетом полученной информации о «плохих-некачественных» узлах было необходимо увеличивать число ячеек до допустимого значения. Относительно небольшое количество ячеек на поверхности тела имели вогнутость или выпуклость, к тому же могли иметь неравные стороны. Проблема устранялась путём выстраивания отдельного слоя ячеек на поверхности тела. В результате полученная сетка имела минимальное искривление геометрии крыла треугольной формы и обеспечивалось устойчивое решение. Более точное разрешение геометрии тела приводило к росту числа элементов сетки, а соответственно к росту вычислительных ресурсов.

Пространственная область для варианта с масштабом 1:1 представляла из себя прямоугольную область с размерами 30м х 10м х 5м. Конфигурационный файл snappyHexMeshDict использовался для задания необходимых параметров и содержал 5 основных секций:

- geometry – описание геометрии;
- castellatedMeshControls – поверхностное и объемное сгущение сетки;
- snapControl – контроль сетки у поверхности тела;

– addLayersControls – контроль и создание ячеек сетки в пограничных слоях;
– meshQualityMetrics – контроль качества сетки по различным индикаторам.

На рис. 3, 4, 5 показаны фрагменты сгенерированной сетки: треугольное крыло и границы расчетной области. Было построено три варианта сетки: 1.1 млн., 3.8 млн., 9.6 млн. ячеек. Количество ячеек в пограничном слое составляло примерно от 400 000 до 3 млн. в зависимости от сетки.

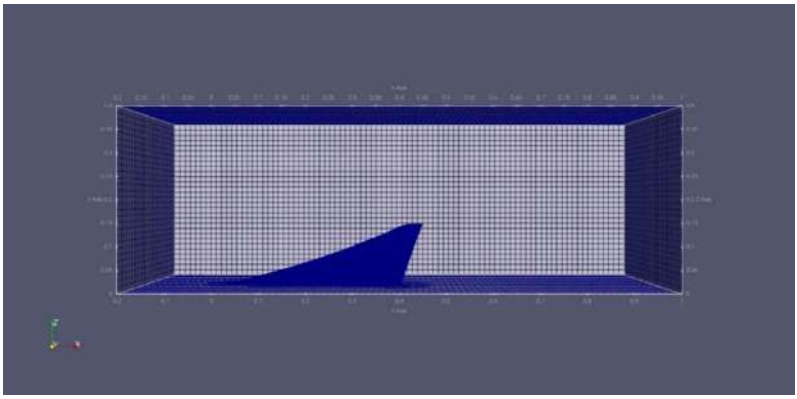


Рис.3Вид на крыло сбоку
Fig.3 Side view of the wing

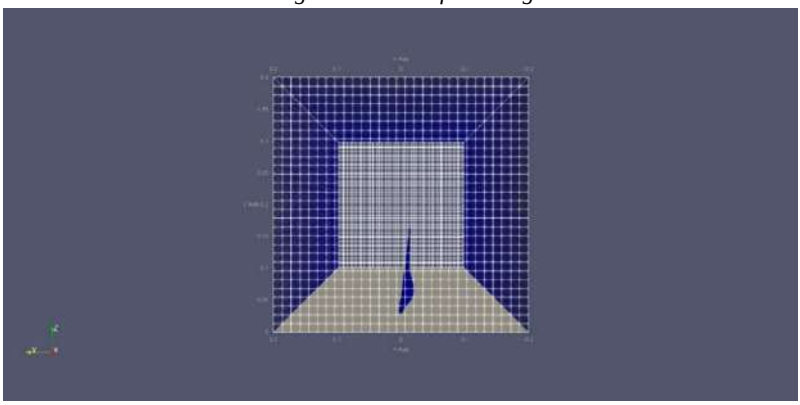


Рис.4 Вид на крыло спереди
Fig.4 Front view of the wing

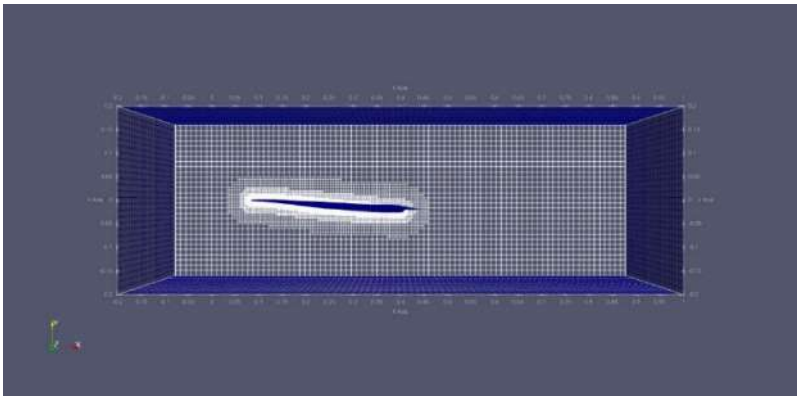


Рис.5 Вид на крыло сверху
Fig. 5 View of the wing from above

После построения сетки целесообразно было проводить контроль качества поверхности.

В секции "geometry" можно было задавать геометрию, название исследуемого тела и формат файла STL, OBJ.

Также возможно определять вспомогательные простые геометрические тела такие как: прямоугольник, сфера. Они необходимы для сгущения сетки вблизи исследуемого тела.
В этом случае необходимо было задавать координаты прямоугольника или координату центра и радиуса сферы.

В секции castellatedMeshControls разбиение ячеек выполнялось в соответствии со спецификацией, установленных пользователем.

Во входном сечении расчетной области задавались "жесткие" граничные условия, на поверхности крыла – условия прилипания с применением пристеночных функций для параметров турбулентности k-ω SST, на остальных сечениях – "мягкие" граничные условия.

5. Алгоритм для работы с ячейками

В результате улучшения качества поверхности искривление геометрической формы исследуемого тела стало минимальным. Под "плохими-некачественными" узлами будем понимать узлы ячеек или граней с большими углами между гранями (более 70°), неверными объемами ячеек и т.п.

Разработанный алгоритм выявления "плохих-некачественных" узлов основан на данных утилиты checkMesh и состоит из следующих основных этапов.

1) Создается список узлов пленки, граничащих с эйлеровой сеткой.
2) Генерируется список узлов пленки, не граничащих с эйлеровой сеткой.
3) Считываются из всех файлов каталога для пленки номера "плохих-некачественных" граней в общий список.
4) Из списка номеров "плохих-некачественных" граней удаляются повторяющиеся.
5) На основе списка номеров "плохих-некачественных" граней формируется список номеров "плохих-некачественных" узлов области пленки, граничащих с эйлеровой сеткой.
6) Из списка номеров, полученного в предыдущем пункте, удаляются повторяющиеся.
7) Формируется список номеров "плохих-некачественных" узлов области пленки, не граничащих с эйлеровой сеткой, но имеющих ребро с "плохими- некачественных" узлами. Они указаны в списке, полученном в п. 5.
8) Из списка номеров, полученного в предыдущем пункте, удаляются повторяющиеся.
9) Удаляются из списка узлов пленки из п. 1 все "плохие-некачественные" узлы с номерами, полученными в п. 6. Таким образом, остаются только "хорошие-качественные" узлы пленки, граничащие с эйлеровой сеткой.
10) Удаляются из списка узлов пленки из п.2 все "плохие-некачественные" узлы с номерами, полученными в п. 8.

Таким образом, только "хорошие-качественные" узлы пленки, будут при необходимости подвергаться изменению своих координат при нарастании льда.

Приведенный алгоритм выполняется только один раз перед началом расчета.

6. Результаты

Были выполнены расчеты обтекания крыла самолета Х-59 в масштабе 1:25 и 1:1. В обоих случаях температура набегающего потока и капелек была равна 261.87 K, MVD=20 микрон, что соответствовало режиму рыхлого льда (rime ice) [4]. При моделировании такого температурного режима можно использовать форсаж времени, т.е. уменьшение времени процесса обледенения за счет увеличения водности без существенных отличий в результатах.

В данных расчетах водность LWC задавалась равной 13.6 г/м^3, а время процесса обледенения 1.8 секунд. Этот режим можно рассматривать как аналогичный режиму с водностью 0.68 г/м^3 и времени обледенения 36 секунд. Значение скорости для набегающего газокапельного потока задавалась равной 90 м/с. Угол атаки задавался равным 0 градусов.

При моделировании процесса обледенения крыла в масштабе 1:1 давление набегающего потока через входное сечение задавалось равным 10^5 Па, а вариант в масштабе 1:25 для сохранения подобия чисел Рейнольдса и Маха давление задавалось в 25 раз больше. Шаг по времени в зависимости от характерных размеров ячеек сетки варьировалось от $2 \cdot 10^{-5}$ с до $2 \cdot 10^{-4}$ с. Вычисления выполнялись на кластере ИСП РАН с использованием 48 или 96 ядер процессоров.

На рис. 6 представлено типичное положение "плохие-некачественные" узлов. Отметим, что узлы на задней кромке на самом деле не представляют проблемы, поскольку для рассчитываемых случаев обледенения там отсутствует пленка воды, а при нулевой толщине пленки воды форма неудачной ячейки не влиятельна. Наличие "плохих-некачественных" узлов на передней кромке крыла, наоборот, говорит о настоятельной необходимости улучшения сетки.

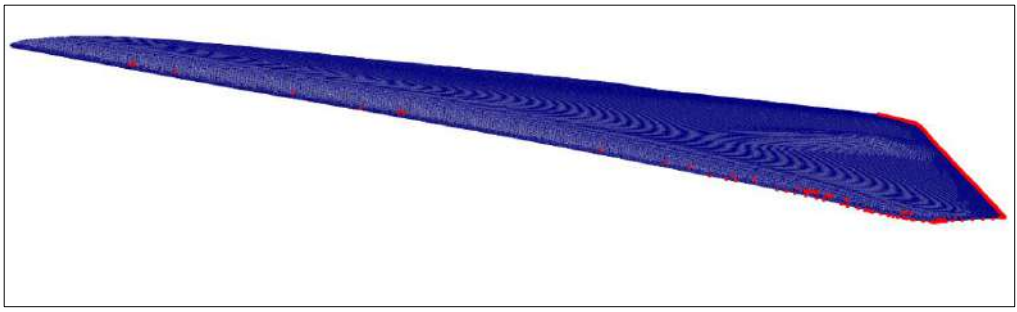


Рис. 6. Визуализация типичного положения «плохих» узлов, выделенных красным цветом
Fig. 6. Visualization of the typical position of "bad" nodes highlighted in red

На рис. 7, 8, 9 представлены сетки для 1.1 миллиона, 3.8 миллиона, 9.6 миллиона ячеек. На рис. 7 заметны деформации граней (вмятость) элементов сетки, находящихся в зоне между верхней и нижней поверхностями крыла (лобовой части). Использование алгоритма выявления «плохих-некачественных» узлов позволило существенно улучшить качество элементов сетки в процессе генерации с помощью утилиты snappyHexMesh, что продемонстрировано на рис.7 ,8, 9.

При расчетах на сетке с 9.6 миллионами ячеек выявились определенные проблемы производительности процесса декомпозиции. Применение параллельных технологий MPI для решателя iceFoam предполагает использования одного и того же ядра процессора для ячеек газовой фазы и пленки, имеющих общую грань.

Такой подход обеспечивает надежность параллельных расчетов при движении границы сетки в ходе вычислений. С этой целью была разработана утилита extrudeToFilmCellDist. В текущем варианте время работы данной утилиты пропорционально n^2, где n - количество ячеек пленки. Для относительно небольших сеток утилита extrudeToFilmCellDist выполнялась в течения максимум нескольких минут. Однако для сетки 9.6 миллионами ячеек время ее выполнения составило несколько часов.

В связи с этим возникла необходимость переработки текста утилиты extrudeToFilmCellDist таким образом, чтобы время ее работы было пропорционально $n \cdot \log(n)$ [8,9].

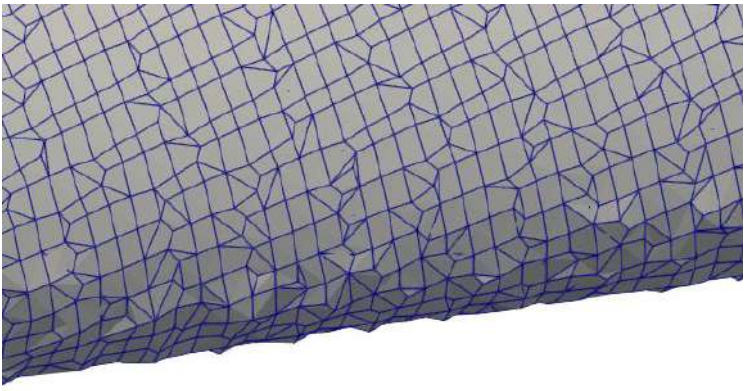


Рис. 7. Фрагмент сетки для случая 1.1 млн ячеек
Fig. 7. Grid fragment for the case of 1.1 million cells

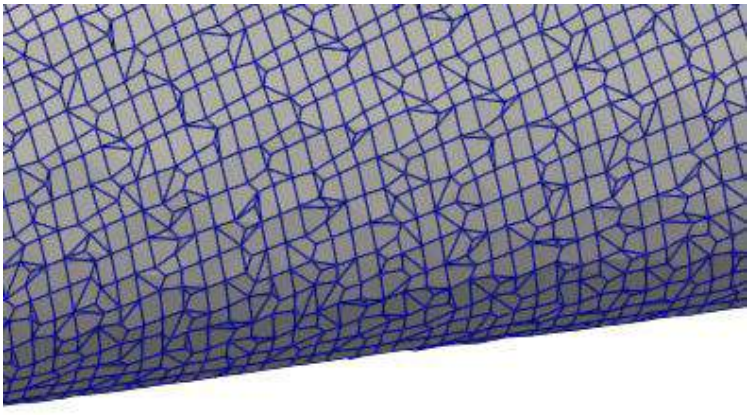


Рис. 8. Фрагмент сетки для случая 3.8 млн ячеек
Fig. 8. Grid fragment for the case of 3.8 million cells

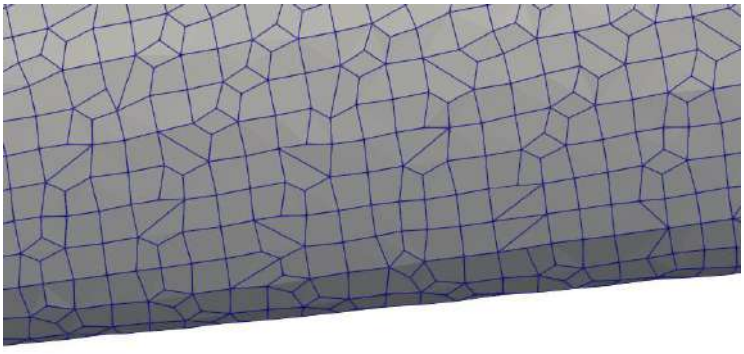


Рис. 9. Фрагмент сетки для случая 9.6 млн ячеек
Fig. 9. Grid fragment for the case of 9.9 million cells

Полученные результаты иллюстрируют положение льда на поверхности исследуемого крыла (синим цветом) для модели 1:25 и давления в 25 атм. (рис.10, 11).

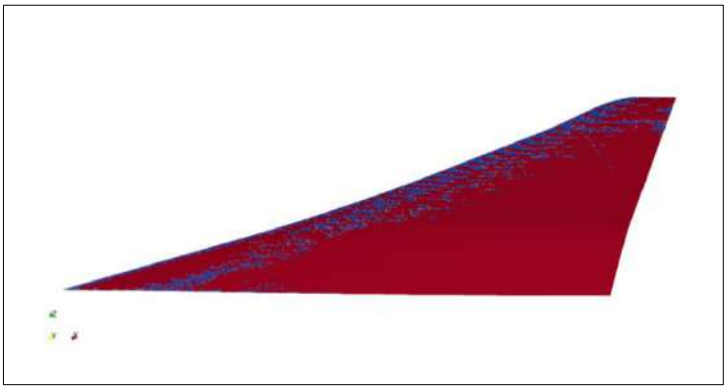


Рис. 10. Положение льда на нижней поверхности треугольного крыла
Fig. 10. Ice location on the lower surface of the delta-wing

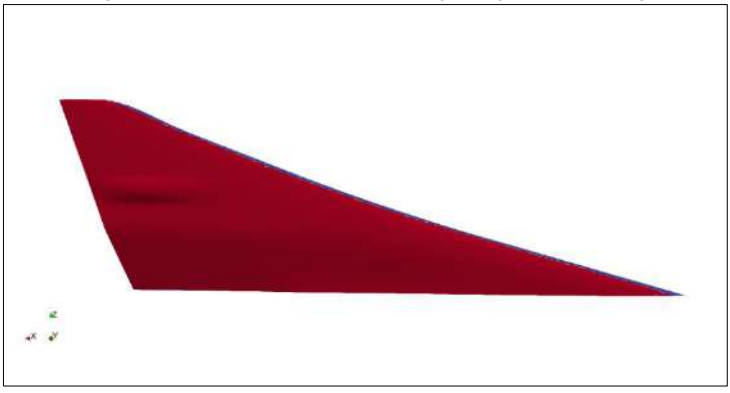


Рис. 11. Положение льда на верхней поверхности треугольного крыла
Fig. 11. Ice location on the upper surface of the delta-wing

Типичное распределение капель воды около крыла представлено на рис. 12. Красноватые оттенки соответствуют положительным значениям вертикальной скорости капель, зеленые и синие – отрицательным значениям.

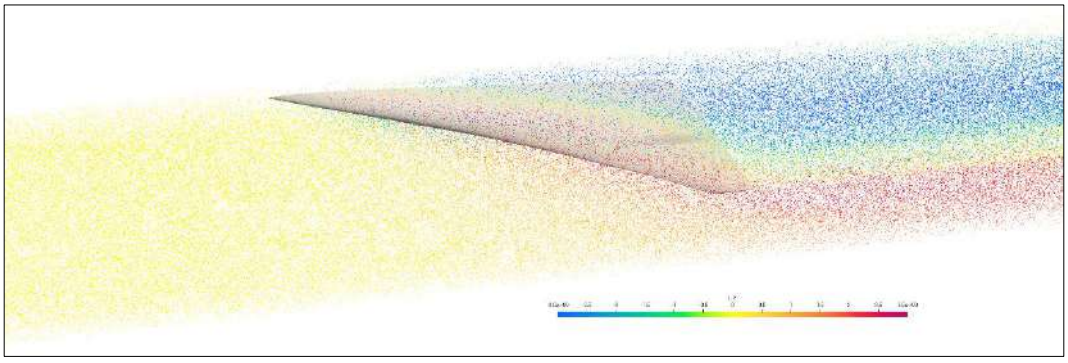


Рис. 12. Распределение капель воды около крыла
Fig. 12. Distribution of water droplets around the wing

В результате вычислений для модели в масштабе 1:25 при давлении 1 атм. наблюдались ручейки воды, а для модели 1:25 и давления 25 атм., а также для модели в масштабе 1:1 ручейков воды не было. Т.е. снизу капельки прямо попадали на плоскость крыла и там замерзали. На рис.13, 14 более детально видна разница в положении областей обледенения

на нижней поверхности фрагмента крыла для разномасштабных моделей. Масса льда для модели масштаба 1:1 составила 72.3 грамма.

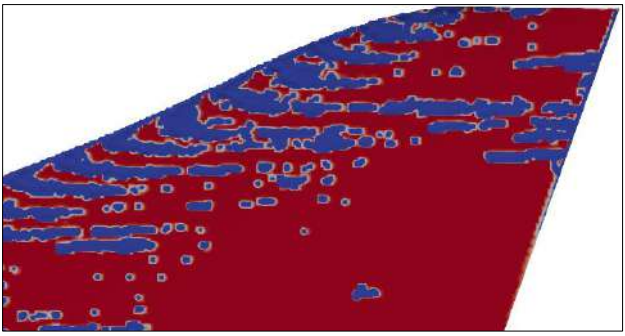


Рис.13. Положение льда на нижней поверхности фрагмента крыла для модели в масштабе 1:25
Fig.13. Ice location on the lower surface of the delta-wing fragment for the 1:25 scale model

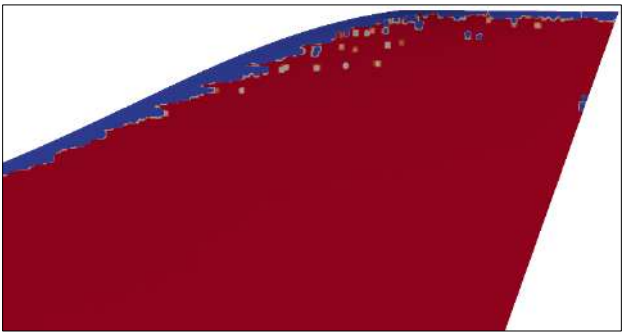


Рис.14. Положение льда на нижней поверхности фрагмента крыла для модели в масштабе 1:1
Fig.14. Ice location on the lower surface of the delta-wing fragment for the 1:1 scale model

Предварительный вывод – даже совпадения чисел Рейнольдса и Маха для несущей фазы недостаточно для разномасштабных моделей обледенения. Необходимо проводить дополнительные исследования с разными масштабами крыла, как это было сделано в работе [10].

7. Заключение

Построение качественных сеток крайне необходимо для задачи моделирования процесса обледенения крыла сложной формы с использованием решателя iceFoam в рамках пакета OpenFOAM. Предложенный способ построения таких сеток на основе утилит snappyHexMesh, extrudeToRegionMesh и алгоритма выявления "плохих-некачественных" узлов сетки пленки позволяет значительно ускорить процесс генерации сеток с требуемым качеством, необходимых для решения данной задачи. Также показаны трудности соблюдения безразмерных параметров, которые могут возникнуть при попытке пространственного масштабирования крыла. Отметим, что применение метода математического моделирования позволяет избежать этих проблем за счет проведения, пусть и более трудоемких, вычислений полномасштабной модели.

Список литературы / References

[1] Low-Boom Flight Demonstration. URL: https://www.nasa.gov/sites/default/files/atoms/files/low-boom-litho-updated-jan2020.pdf, accessed 09 October 2022.

[2] ASA 2022 Special Focus Boom Session. URL: https://lbpw.larc.nasa.gov, accessed 09 October 2022.

[3] Аэродинамика, устойчивость и управляемость сверхзвуковых самолетов. Под ред. Бюшгенса Г.С. М., Наука, Физматлит, 1998 г., 816 стр. / Aerodynamics, stability and controllability of supersonic aircraft. Ed. Byushgens G.S. M., Nauka, Fizmatlit, 1998, 816 p. (in Russian).

[4] Papadakis M., Yeong H. et al. Experimental Investigation of Ice Accretion Effects on a Swept-Wing. Technical Report DOT/FAA/AR-05/39, NASA, 2005.

[5] Кошелев К.Б., Мельникова В.Г., Стрижак С.В. Разработка решателя iceFoam для моделирования процесса обледенения. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 217–234 / Koshelev K.B., Melnikova V.G. Strijhak S.V. Development of iceFom solver for modeling ice accretion. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 217–234 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–16.

[6] Strijhak S., Ryazanov D. et al. A Neural Network Prediction for Ice Shapes on Airfoils Using iceFoam Simulations. Aerospace, vol. 9, issue 2, 2022, article no. 96, 28 p.

[7] Bourgault Y., Beaugendre H., Habashi W.G. Development of a Shallow-Water Icing Model in FENSAP-ICE. Journal of Aircraft, vol. 37, issue 4, 2000, pp. 640–646.

[8] Гергель В.П., Сысоев А.В. Высокопроизводительные параллельные вычисления. 100 заданий для расширенного лабораторного практикума. М., Физматлит, 2018 г., 248 стр. / [8]. Gergel V.P., Sysoev A.V. High performance parallel computing. 100 tasks for an extended laboratory practice. M., Fizmatlit, 2018, 248 p. (in Russian).

[9] Якобовский М.В. Введение в параллельные методы решения задач. М., Изд-во МГУ, 2013 г., 328 стр. / Yakobovsky M.V. Introduction to parallel methods for solving problems. M., Publishing House of Moscow State University, 2013, 328 p. (in Russian).

[10] Fujiwara G.E.C., Bragg M.B., Broeren A.P. Comparison of Computational and Experimental Ice Accretions of Large Swept Wings. Journal of Aircraft vol. 57, issue 2, 2020, pp. 342-359.

Информация об авторах / Information about authors

Константин Борисович КОШЕЛЕВ – кандидат физико-математических наук, доцент, старший научный сотрудник. Сфера научных интересов: вычислительная гидродинамика, гидрология, геоинформатика.

Konstantin Borisovich KOSHELEV is candidate of physical and mathematical sciences, associate professor, senior researcher. Research interests: computational fluid dynamics.

Андрей Владимирович ОСИПОВ – инженер. Сфера научных интересов: вычислительная гидродинамика, метод контрольного объема, подвижные сетки, лагранжев подход.

Andrei Vladimirovich OSIPOV – engineer. Research interests: computational fluid dynamics, finite volume method, dynamic meshes, particles.

Сергей Владимирович СТРИЖАК – кандидат технических наук, ведущий инженер. Сфера научных интересов: вычислительная гидродинамика, многофазные течения, турбулентность, ветроэнергетика, параллельные вычисления.

Sergei Vladimirovich STRIJHAK – candidate of technical sciences, leading engineer. Research interests: computational fluid dynamics.

DOI: 10.15514/ISPRAS-2022-34(5)-16

Математическое моделирование гидродинамических процессов в прибрежной акватории Японского моря

[1,2] *Е. В. Амосова, ORCID: 0000-0003-2154-5010 <el_amosova@mail.ru>*
[1,2] *К. С. Кузнецов, ORCID: 0000-0002-8204-6138 <kuznetsovks17@gmail.com>*
[2] *В. С. Лемешев, ORCID: 0000-0002-9705-9388 <lemeshev.vs.306@gmail.com>*

[1] *Институт прикладной математики ДВО РАН,*
690041, Россия, г. Владивосток, ул. Радио, д. 7
[2] *Дальневосточный Федеральный Университет,*
690922, Россия, г. Владивосток, о. Русский, п. Аякс, 10

Аннотация. Представлена трехмерная математическая модель динамики прибрежных вод в заливе Петра Великого Японского моря. Система уравнений Навье-Стокса записана в предположении гидродинамического приближения неоднородной жидкости учитывает, турбулентный обмен, ветровые напряжения, силы трения, сложный рельеф дна и береговой линии. На основе гидрографической информации используется численный метод восстановления рельефа дна и береговой линии. Уравнение свободной поверхности воды заменяется сингулярно возмущенной задачей. В прикладном программном пакете FreeFem++ написан код для расчёта гидродинамических характеристик прибрежной акватории. В ходе решения сетка конечных элементов строится автоматически, параметры сетки задаются пользователем. Для решения сингулярной задачи на каждом временном слое происходит адаптация сетки вокруг численного решения с параметрами адаптации сетки.

Ключевые слова: математическое моделирование; метод конечных элементов; уравнения мелкой воды; FreeFem++

Для цитирования: Амосова Е.В., Кузнецов К.С., Лемешев В.С. Математическое моделирование гидродинамических процессов в прибрежной акватории Японского моря. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 227-242. DOI: 10.15514/ISPRAS-2022-34(5)-16

Благодарности: Работа выполнена при финансовой поддержке ДВФУ (Программа стратегического академического лидерства «Приоритет-2030»: Центр цифрового развития).

Mathematical modeling of hydrodynamic processes in the coastal waters of the Sea of Japan

[1,2] *E. V. Amosova, ORCID: 0000-0003-2154-5010 <ev_amosova@mail.ru>*
[1, 2] *K. S. Kuznetsov, ORCID: 0000-0002-8204-6138 <kuznetsovks17@gmail.com>*
[2] *V. S. Lemeshev, ORCID: 0000-0002-9705-9388 <lemeshev.vs.306@gmail.com>*

[1] *Institute of Applied Mathematics of the Far Eastern Branch of the Russian Academy of Sciences,*
7 Radio st., Vladivostok, Russia, 690041
[2] *Far Eastern Federal University,*
10 Ajax Bay, Russky Island, Vladivostok, Russia, 690092

Abstract. A three-dimensional mathematical model of coastal water dynamics in the Peter the Great Bay of the Sea of Japan is presented. The system of Navier-Stokes equations is written under the assumption of a hydrodynamic approximation of an inhomogeneous fluid and takes into account turbulent exchange, wind

stresses, friction forces, complex topography of the bottom and coastline. On the basis of hydrographic information, a numerical method is used to reconstruct the bottom topography and coastline. The free water surface equation is replaced by a singularly perturbed problem. In the FreeFem++ application software package, a code was written for calculating the hydrodynamic characteristics of coastal waters. During the solution, the finite element mesh is built automatically, the mesh parameters are set by the user. To solve the singular problem, at each time layer, the grid is adapted around the numerical solution with the grid adaptation parameters.

Keywords: math modeling; finite element method; shallow water equations; FreeFem++

For citation: Amosova E.V., Kuznetsov K.S., Lemeshev V.S. Mathematical modeling of hydrodynamic processes in the coastal waters of the Sea of Japan. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 227-242 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-16

Acknowledgements. This work was supported by Far Eastern Federal University (or FEFU) (Program "PRIORITY 2030": Digital Science).

1. Введение

Проблема негативного антропогенного влияния на водную среду, и, как следствие, загрязнение морских зон в результате хозяйственной деятельности человека, является актуальной на сегодняшний день. Особо остро она стоит в приморских регионах, где количество акваторий больше, чем в континентальных районах страны. Одним из важных с экономической точки зрения регионов является Дальний Восток. В последнее время из-за роста добывающей и обрабатывающей промышленности в прибрежной зоне проводятся многие виды работ, таких как строительство буровых платформ, ремонтные и восстановительные работы в портовых территориях, дноуглубительные работы при строительстве, прокладка различных трубопроводов. Вследствие этого выброс загрязняющих веществ в море возрастает.

Для снижения уровня загрязнения необходима точная оценка воздействия деятельности человека на гидросферу. Прогнозирование формирования фронта загрязнения и его распространение – задача, решение которой заключается в грамотном моделировании процессов, происходящих при проведении работ в акваториях. Исследование данной проблемы и моделирование распространения загрязняющих веществ представляют большой интерес и активно развиваются.

Для сохранения биоресурсов и проведения восстановительных мероприятий, призванных улучшить экологическую обстановку, разработана нормативно-правовая база [1-3]. Однако, для прогнозирования качества окружающей среды, требуется проведение математического моделирования процессов распространения загрязнений в акваториях.

На данный момент известно большое число различных математических моделей, описывающих процессы переноса в водоемах и подземных водах. Многие отечественные и иностранные ученые посвятили свои труды проблематике моделирования распространения взвесей [4-10]. Однако, несмотря на очевидную актуальность и длительное исследование проблемы, не существует единой модели, описывающей все возможные сценарии распространения загрязняющей массы в различных акваториях. Связано это с особенностями рассматриваемого региона, геометрией загрязнения, различной степенью влияния факторов на распространение загрязнения и сложностью моделирования процесса турбулентности, который в свою очередь так же не имеет единой методики описания.

В работе представлено математическое моделирование гидродинамических процессов в прибрежных водах Японского моря, основанное на трехмерных уравнениях Навье-Стокса, учитывающих гидростатическое приближение. Изменения температуры и солености морской воды описываются похожими уравнениями диффузии. В них через балансы тепла и влаги учитывается влияние атмосферы, которое распространяется от поверхности в толщу

воды, и особенно заметно от сезона к сезону. При изучении плотностной устойчивости вод и их циркуляции учитывается совместное влияние как температуры, так и солености. Действие внутренних источников и стоков тепла за счет фазовых преобразований воды и диссипации механической энергии незначительно. Наиболее существенно на изменение температуры влияет испарение воды. Это поверхностный процесс, поэтому он учитывается не в уравнении теплопроводности, а в уравнении теплового баланса поверхности океана, которое используется в качестве одного из граничных условий к уравнению теплопроводности. Уравнение баланса соли поверхности океана представляет собой сумму потоков соли, обусловленных притоком или оттоком массы пресной воды за счет осадков на границе и испарением. Считая, что вклад теплового потока от дна незначителен, на дне водоема задаются значения солености и температуры, соответствующие значениям на данных уровнях морской акватории. Потоки соли на дне водоема отсутствуют. Данные для начальных условий рассчитываются из соответствующей стационарной модели. На основе гидрографической информации используется численный метод восстановления рельефа дна и береговой линии, что важно для прибрежных районов.

2. Постановка задачи

Задачи рассматривается в трехмерной ограниченной области. Граница области состоит из двух горизонтальных поверхностей: дно акватории $h(x,y) > 0$ и свободная поверхность – функция $\xi(x,y,t)$ определяется в процессе решения, вертикальная граница содержит участок входа воды в область Γ_{in}, на котором задается вектор скорости, равный скорости дрейфового потока и уровень воды, соответствующий уровню прилива или отлива. На оставшейся части границы задаются условия либо трения стенки, либо условия непроницаемости.

Пусть $Q \subset R^3$ – ограниченная область, ∂Q – граница области, такая что $\partial Q = \Omega_h \cup \Omega_\xi \cup \Gamma$, где

$$\Omega_h = \{(x,y): z = -h(x,y)\}, \qquad \Omega_\xi = \{(x,y): z = \xi(x,y,t)\}.$$

Вертикальная граница $\Gamma = \Gamma_{in} \cup \Gamma_{out}$. Под Γ_{in} будем понимать границу входа, под Γ_{out} – свободную границу. Область изменений переменных (x,y) для каждого z обозначим через Ω, тогда $Q = \Omega \times [-h,\ \xi(t)]$.

Пусть $\mathrm{v} = \{u(x,y,z,t);\ v(x,y,z,t);\ w(x,y,z,t)\}$ – вектор скорости движения рассматриваемой среды, $p = p(x,y,z,t)$ – давление среды, $\rho = \rho(x,y,z,t)$ – плотность среды, $T = T(x,y,z,t)$ – температура среды, $S = S(x,y,z,t)$ – соленость среды.

Под осредненным по вертикали значением некоторой функции f будем понимать следующее выражение:

$$\bar{f}(x,y,t) = \frac{1}{H}\int_{-h}^{\xi} f(x,y,z,t)dz\,, \tag{1}$$

где

$$H(x,y,t) = h(x,y) + \xi(x,y,t). \tag{2}$$

Из-за турбулентного характера движения морской воды нет практического смысла находить мгновенные значения характеристик среды. Обычно проводят осреднение искомых величин в некотором диапазоне времени. В результате операции осреднения по времени трехмерные уравнения движения, записанные в гидростатическом приближении [11], уравнение несжимаемости, температуры и солености примут следующий вид [11]:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=-\frac{1}{\rho_0}\frac{\partial p}{\partial x}+K\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\frac{\partial}{\partial z}\left(\mu_t\frac{\partial u}{\partial z}\right)+f_c v,$$
$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}=-\frac{1}{\rho_0}\frac{\partial p}{\partial y}+K\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+\frac{\partial}{\partial z}\left(\mu_t\frac{\partial v}{\partial z}\right)-f_c u,$$
$$\frac{\partial p}{\partial z}=-\rho g,\ \frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0, \tag{3}$$
$$\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}+w\frac{\partial T}{\partial z}=K_T\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+\frac{\partial}{\partial z}\left(\nu_t\frac{\partial T}{\partial z}\right),$$
$$\frac{\partial S}{\partial t}+u\frac{\partial S}{\partial x}+v\frac{\partial S}{\partial y}+w\frac{\partial S}{\partial z}=K_T\left(\frac{\partial^2 S}{\partial x^2}+\frac{\partial^2 S}{\partial y^2}\right)+\frac{\partial}{\partial z}\left(\nu_t\frac{\partial S}{\partial z}\right),$$
$$(x,y)\in\Omega,\ -h(x,y)\le z\le\xi(x,y,t).$$

Здесь g – ускорение свободного падения, f_c – параметр Кориолиса, ρ_0 – среднее значение плотности морской воды. Плотность среды ρ определяется по температуре и солености жидкой среды. Для расчета принята формула, полученная из уравнения состояния по форме О. И. Мамаева:

$$\rho=\rho_0(1-\alpha_1 T-b_1T^2+c_1S-d_1TS) \tag{4}$$

где

$$\rho_0=1.00007,\quad \alpha_1=0.3449\cdot10^{-5},\quad b_1=0.4689\cdot10^{-5},\quad c_1=80.1943\cdot10^{-5},$$
$$\alpha_1=0.2000\cdot10^{-5}.$$

Уравнение для давления разрешается следующим образом:

$$p=P_a+g\rho_0\xi+g\int_0^z\rho dz',\quad (x,y)\in\Omega,\quad -h(x,y)\le z\le 0. \tag{5}$$

При моделировании коэффициента турбулентности учитывается анизотропность турбулентного обмена. Вследствие преобладания горизонтального масштаба над вертикальным, несущественна роль горизонтальной турбулентной вязкости в сравнении с вертикальной. Достаточно малый масштаб вертикального разрешения области позволяет использовать формулу Смагоринского для расчета вертикального коэффициента турбулентности:

$$\mu_t=(C_s\Delta)^2 s^{\frac{1}{2}}, s=2s_{ij}\cdot s_{ij}, \tag{6}$$

где s_{ij} – осредненный тензор скоростей деформаций, C_s – безразмерная эмпирическая константа, Δ – характерный масштаб сетки, $\nu_t=\mu_t$. В работе приняты значения горизонтального коэффициента турбулентности и коэффициента диффузии равными, $K_T=$ К $=10^3$ *м*^{2}/*с* .

Из закона сохранения массы для конечного объема в каждый момент времени, учитывая (1) и (2), получим уравнение для определения функции уровня поверхности:

$$\frac{\partial(\bar\rho H)}{\partial t}+u\frac{\partial(\bar\rho H)}{\partial x}+v\frac{\partial(\bar\rho H)}{\partial y}+w\frac{\partial(\bar\rho H)}{\partial z}+\bar\rho H\left(\frac{\partial\bar u}{\partial x}+\frac{\partial\bar v}{\partial x}\right)=0,\quad (x,y)\in\Omega. \tag{7}$$

Система уравнений (2) – (7) рассматривается при следующих граничных условиях.

На поверхности акватории Ω_ξ учитываются ветровые напряжения **St**, которые описываются с помощью квадратичного закона изменения скорости ветра

$$\mathbf{St}=-\frac{\rho_a}{\rho_0}C_d\mathbf{W}|\mathbf{W}|,$$

где ρ_a – плотность ветра, **W** – вектор-скорости ветра на уровне 10 *м* над свободной поверхностью, C_d – коэффициент трения, который определяется следующим образом:

$$C_d = \begin{cases} 0.564 \cdot 10^{-3}, & \text{если } \mathbf{W} \le 4.917; \\ (-0.12 + 0.13 \cdot \mathbf{W}) \cdot 10^{-3}, & \text{если } 4.917 \le \mathbf{W} \le 19.221; \\ 2.514 \cdot 10^{-3}, & \text{если } \mathbf{W} > 19.221. \end{cases}$$

Условия трения на поверхности заданы с помощью тензора напряжений:

$$\mathbf{v} \cdot \mathbf{n} = 0, \qquad (\mu D(\mathbf{v})\mathbf{n} - \mathbf{n}p) = -\mathbf{St}, \qquad z = \xi(x, y, t). \tag{8}$$

Здесь $\mathbf{n}$ – вектор внешней нормали, $D(\mathbf{v})$ – тензор напряжений, μ – матрица коэффициентов турбулентности, $\mu_{ij} = K$, $i, j = 1, 2$, $\mu_{l,3} = \mu_t$, $l = 1, 2, 3$.

На поверхности моря $z = \xi(x, y, t)$ используется уравнение баланса тепла и соли в безадвективном районе океана:

$$\lambda \frac{\partial T}{\partial z} n_z = c_v \rho_a C_T |W| \left(T_g - T_0\right) + L_u \rho_a C_q |W| \left(q_g - q_0\right) + \Phi, \quad \frac{\partial T}{\partial x} = \frac{\partial T}{\partial y} = 0,$$

$$\nu_s \frac{\partial S}{\partial z} n_z = S_0 \frac{\rho_a}{\rho_0} C_q |\boldsymbol{W}| (q_a - q_0) + \frac{S_0}{\rho_0} \frac{\partial M}{\partial t}, \qquad \frac{\partial S}{\partial x} = \frac{\partial S}{\partial y} = 0. \tag{9}$$

Здесь T_0 – температура на поверхности воды, T_g – температура окружающей среды, q_a – удельная влажность воздуха, q_0 – удельная влажность морской воды, S_0 – средняя соленость, $M(t)$ – масса осадков, λ – коэффициент теплопроводности соленой воды, $\lambda = K_T \rho_0 c_v$, c_v – теплоемкость соленой воды, $C_T = 2.2 \cdot 10^{-3}, C_q = 2.7 \cdot 10^{-3}$ – коэффициенты трения, Ф – поток тепла, обусловленный радиацией, испарением и турбулентной теплоотдачей.

Напряжения трения на дне акватории Ω_h также задается с помощью квадратичного закона трения

$$\mathbf{v} \cdot \mathbf{n} = 0, \qquad (\mu D(\mathbf{v})\mathbf{n} - \mathbf{n}p) = -f_b \mathbf{v}|\mathbf{v}|, \qquad z = -h(x, y),$$

где f_b – коэффициент придонного трения.

В задачах моделирования гидродинамических процессов для определения коэффициента трения часто используется соотношение, из которого следует, что величина коэффициента трения уменьшается с увеличением глубины, что накладывает существенное ограничение применимости данной формулы при моделировании прибрежной бухты. В работе для определения коэффициента трения принята формула

$$f_b = \frac{g n^2}{H_{max}^{4/3}} |\mathbf{v}|,$$

где n −параметр шерховатости, $H_{max} = \max\limits_{(x,y) \in \Omega} |h(x, y)|$.

Для вертикальной составляющей компоненты вектора скорости задается кинематическое условие на дне моря

$$w|_{z=-h(x,y)} = -u\left(x, y, -h(x, y)\right) \frac{\partial h}{\partial x} - v\left(x, y, -h(x, y)\right) \frac{\partial h}{\partial y}. \tag{10}$$

Значения температуры и солености на дне акватории соответствуют уровню глубины для данного времени года:

$$T|_{z=-h(x,y)} = T_h(x, y), \qquad S|_{z=-h(x,y)} = S_h(x, y).$$

На участке входа Γ_{in} задается уровень поверхности воды (приливы – отливы) и скорость течения:

$$\xi|_{\Gamma_{in}} = \xi_g(t), \qquad u|_{\Gamma_{in}} = u_g + u_s(t), \qquad v|_{\Gamma_{in}} = v_g + v_s(t), \qquad w|_{\Gamma_{in}} = w_g + w_s(t). \tag{11}$$

В качестве стационарной части вектора скорости на входе принимается скорость, найденная из уравнений динамики воды, записанных для установившегося дрейфового течения:

$$\mu_t \frac{\partial^2 u_g}{\partial z^2} = -f_c v_g, \qquad \mu_t \frac{\partial^2 v_g}{\partial z^2} = -f_c u_g,$$

с краевыми условиями:

$$-\mu_t \frac{\partial u_g}{\partial z}\Big|_{z=\xi} = \mathbf{St}_x, \qquad -\mu_t \frac{\partial v_g}{\partial z}\Big|_{z=\xi} = \mathbf{St}_y,$$
$$u_g\big|_{z=-\infty} \neq \infty, \qquad v_g\big|_{z=-\infty} \neq \infty. \tag{12}$$

Компонента w_g определяется из уравнения неразрывности и условия (10) при значениях компонент вектора скорости $u = u_g,\ v = v_g$.

На свободной границе Γ_{out} считаем, что все потоки либо отсутствуют, либо для скорости задан квадратичный закон трения (8), а тепловые и солевые потоки равны нулю.

В начальный момент времени задан уровень поверхности акватории ξ_0, по которому определяется условие для выражения $(\bar{\rho}H)$ в начальный момент времени:

$$(\bar{\rho}H)|_{t=0} = \int_{-h}^{0} \rho_s\, dz + \rho_s(x,y,0)\xi_0,\ \ \rho_s = \rho_s(T_0, S_0\ ),$$

где T_0, S_0 – начальное распределение температуры и солености, при этом

$$T|_{t=0} = T_0(x,y,z), \qquad S|_{t=0} = S_0(x,y,z),$$

и вектора-скорости

$$u|_{t=0} = u_0(x,y,z), \qquad v|_{t=0} = v_0(x,y,z), \qquad w|_{t=0} = w_0(x,y,z). \tag{13}$$

Стационарные характеристики (13) определяются из решения соответствующей стационарной задачи (2) – (12).

3. Моделирование области

Поставленная задача решается методом конечных элементов с дискретизацией по времени всех уравнений. Расчёт модели проводится в программном комплексе FreeFem++ методом конечных элементов на заданной прибрежной зоне Японского моря. FreeFem++ – это программа численного решения дифференциальных уравнений в частных производных, является свободно распространяемым программным обеспечением, имеет ограниченно открытую лицензию, основывается на методе конечных элементов и базируется на языке программирования C++. Среда FreeFem++ не включает в себя CAD-систему и не позволяет импортировать геометрию области определения из CAD-систем, поэтому геометрия области определения описываться математическими формулами в самом пакете. В ходе решения сетка конечных элементов строится автоматически, параметры сетки задаются пользователем, возможна адаптация сетки (выбор наиболее подходящего размера элементов для каждой части области определения) для конкретной задачи.

При математическом моделировании гидродинамики морской акватории актуальной является задача обработки гидрографической информации. Глубина водоема и береговая линия прибрежной зоны задаются в отдельных точках. Использование расчетной сетки, построенной на «грубой» геометрии, приводит к локальным всплескам вычисляемых функций и как следствие к большим погрешностям вычислений. Для повышения точности расчетов гидродинамических процессов необходимо приблизить функцию двух переменных, описывающую рельеф дна акватории и береговую линию, более гладкими.

В работе проведен тестовый расчет гидродинамических характеристик для бухты Аякс, расположенной в заливе Петра Великого Японского моря.

На рис. 1 представлен фрагмент карты бухты Аякс залива Петра великого Японского моря с триангуляцией двумерной области, границами которой являются берега бухты. График кусочно-постоянной поверхности, описывающей дно бухты, граница которой построена с помощью одномерной интерполяции по координатам берегов бухты и изолиний соответствующим принятым значениям глубин, представлен на рис. 2.

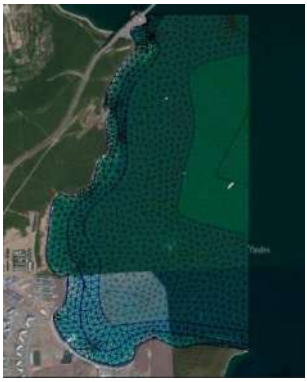


Рис. 1. Бухта Аякс в заливе Петра Великого Японского моря
Fig. 1. Ajax Bay in the Gulf of Peter the Great Sea of Japan

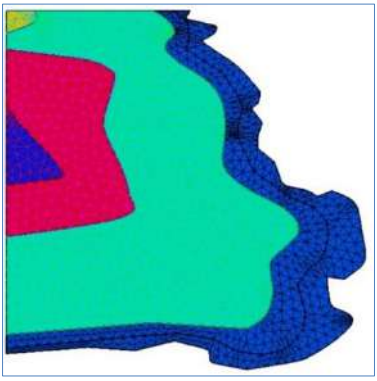


Рис. 2. Поверхность функции дна
Fig. 2. Bottom function surface

В пакете FreeFem++ предусмотрена процедура построения расчетной сетки в случае трехмерной области послойно. Трехмерная область задается как прямое произведение множеств, $Q = \Omega \times [z_{max}; z_{min}]$, где z_{min}, z_{max} – гладкие функции, описывающие соответственно нижнюю и верхнюю границы объема Q.

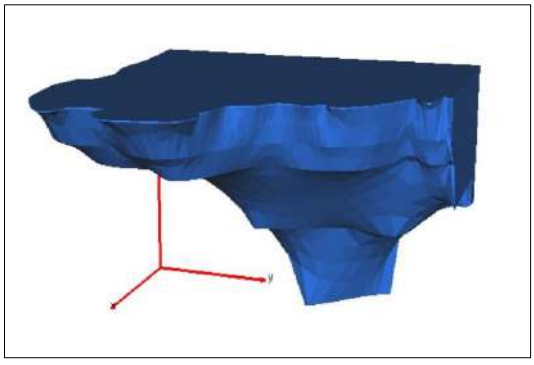


Рис. 3. Трехмерная модель расчетной области
Fig. 3. 3D model of computational domain

Геометрия модели бухты Аякс Японского моря представлена на рис. 3.

4. Обезразмеривание задачи

Пусть t_c, L, H_c, u_c, w_c, T_c, S_c, p_c, ρ_c – характерные масштабы времени, горизонтальной и вертикальной координат, горизонтальной и вертикальной скоростей, температуры, солености, давления, плотности соответственно. Безразмерные величины φ_b введем по формулам:

$$t = t_b t_c, \quad x = x_b L, \quad y = y_b L, \quad z = z_b H_c,$$
$$\nu_s = \nu_{sb}\nu_{sc}, \quad u = u_b u_c, \quad v = v_b u_c, \quad w = w_b w_c, \quad \rho = \rho_b \rho_c, \quad p = p_b p_c,$$
$$T = T_b T_c, \quad S = S_b S_c. \tag{14}$$

Введем следующие безразмерные параметры:

$$Sh = \frac{L}{u_c t_c}, \qquad Eu = \frac{\rho_c u_c^2}{p_c}, \qquad Ki = \frac{u_c}{f_c L},$$

где Sh – число Струхаля, Eu – число Эйлера, Ki – число Кибеля. Обозначим

$$Pe_L = \frac{L u_c}{K_T}, \quad Ps_H = \frac{H_c w_c}{\nu_t}, \qquad E_L = \frac{f_c L^2}{K}, \qquad E_H = \frac{f_c H_c^2}{\mu_t}, \tag{15}$$

где Pe_L, Ps_H – числа Пекле для горизонтальной и вертикальной диффузий, $E_L, E_H -$ числа Экмана для вертикальной и горизонтальной турбулентной вязкости.

Анализ уравнения неразрывности показывает, что произведение величин характерной вертикальной составляющей скорости на характерный горизонтальный масштаб равно произведению величин характерной горизонтальной составляющей скорости на характерный вертикальный масштаб, т. е. характерное значение вертикальной составляющей скорости определяется соотношением

$$w_c = \frac{H_c}{L} u_c. \tag{16}$$

В дальнейшем, при обозначении безразмерных величин φ_b будем опускать индекс b. Перейдем к обезразмериванию задачи (1) – (13), используя (14) – (16).

Отметим, что исходная система дифференциальных уравнений с частными производными (3), (7) является смешанной и состоит из уравнений параболического и гиперболического типов. Уравнение свободной поверхности воды приводится к параболическому типу с помощью замены уравнения (7) соответствующим параболическим уравнением с малым параметром регуляризации, а из уравнения неразрывности выразим вертикальную составляющую вектора скорости путем интегрирования его по вертикальной координате, учитывая кинематическое условие на дне моря (10).

Запишем безразмерную параболическую систему дифференциальных уравнений с краевыми условиями для определения гидродинамических характеристик среды в заданной акватории.

Уравнения для скорости в безразмерных переменных:

$$ShKi\frac{\partial u}{\partial t} + Ki\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right) = -\frac{Ki}{Eu}\frac{\partial p}{\partial x} + v + \frac{K}{E_L}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{\mu_t}{E_H}\frac{\partial^2 u}{\partial z^2},$$
$$ShKi\frac{\partial v}{\partial t} + Ki\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}\right) = -\frac{Ki}{Eu}\frac{\partial p}{\partial y} - u + \frac{K}{E_L}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{\mu_t}{E_H}\frac{\partial^2 v}{\partial z^2},$$
$$w = w|_{z=-h(x,y)} - \int_{-h(x,y)}^{z}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)d\zeta\,, \qquad -h(x,y) < z < \xi(x,y,t). \tag{17}$$

Безразмерная функция давления определяется из условия

$$p = 1 + \xi\frac{gH_c\rho_c}{p_c} + \frac{gH_c\rho_c}{p_c}\int_z^0 \rho(\zeta)d\zeta\,, \qquad -h(x,y) \le z \le 0. \tag{18}$$

где $h(x,y) > 0$ – функция, аппроксимирующая дно, $\xi(x,y,t)$ – функция свободной поверхности акватории.

Осреднение по вертикальной координате функций будем проводить по правилу (1). Обозначим через

$$\Psi = H\bar{\rho}, \qquad H(x,y,t) = h(x,y) + \xi(x,y,t). \tag{19}$$

Для функции Ψ, согласно (7), запишем параболическое уравнение с малым параметром регуляризации $\varepsilon > 0$:

$$Sh\frac{\partial \Psi}{\partial t} + \bar{u}\frac{\partial \Psi}{\partial x} + \bar{v}\frac{\partial \Psi}{\partial y} + \Psi\left(\frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial x}\right) = \varepsilon\Delta\Psi. \tag{20}$$

Уравнения температуры и солености в безразмерной постановке имеют следующий вид:

$$Sh\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + w\frac{\partial T}{\partial z} = \frac{K_T}{Pe_L}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) + \frac{\nu_t}{Ps_H}\frac{\partial^2 T}{\partial z^2},$$
$$Sh\frac{\partial S}{\partial t} + u\frac{\partial S}{\partial x} + v\frac{\partial S}{\partial y} + w\frac{\partial S}{\partial z} = \frac{K_\mathrm{T}}{Pe_L}\left(\frac{\partial^2 S}{\partial x^2} + \frac{\partial^2 S}{\partial y^2}\right) + \frac{\nu_t}{Ps_H}\frac{\partial^2 S}{\partial z^2}. \tag{21}$$

Из формулы (6) в силу малости градиентов вертикальной компоненты вектора скорости, пренебрегая горизонтальными градиентами скорости, получим следующее выражение:

$$\mu_t = C_s^2 h_z^2 \frac{u_c}{2}\sqrt{\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2}, \qquad C_s = 0.2,$$

где h_z – шаг сетки в направлении оси Oz.

Получим выражения для краевых условий, записанных в безразмерном виде.

Пренебрегая изменением горизонтальных градиентов вектора скорости, запишем условия на поверхности акватории $z = \xi(x,y,t)$, $(x,t) \in \Omega$ для вектора скорости в виде

$$\frac{\mu_t}{E_H}\frac{\partial u}{\partial z}n_z - \frac{L}{H_c}\frac{Ki}{Eu}pn_x = \frac{C_d}{H_c f_c u_c}\frac{\rho_a}{\rho_0}W_x|\mathbf{W}|,$$
$$\frac{\mu_t}{E_H}\frac{\partial v}{\partial z}n_z - \frac{L}{H_c}\frac{Ki}{Eu}pn_y = \frac{C_d}{H_c f_c u_c}\frac{\rho_a}{\rho_0}W_y|\mathbf{W}|, \qquad \mathbf{v}\cdot\mathbf{n} = 0. \tag{22}$$

Условия для температуры и солености на поверхности акватории $z = \xi(x,y,t)$, $(x,t) \in \Omega_\xi$ имеют вид:

$$\frac{\lambda}{P_{SH}}\frac{\partial T}{\partial z} = c_v\rho_a C_T\frac{|\mathbf{W}|}{w_c}\left(T_g - T_0\right) + L_u\rho_a C_q\frac{|\mathbf{W}|}{w_c}\left(T_g - T_0\right) + \frac{\Phi}{w_c}, \quad \frac{\partial T}{\partial x} = \frac{\partial T}{\partial y} = 0,$$
$$\frac{\nu_t}{Ps_H}\frac{\partial S}{\partial z} = \frac{S_0\rho_a|\mathbf{W}|}{S_c\rho_0 w_c}C_q(q_a - q_0) + \frac{S_0}{S_c\rho_0 w_c}\frac{\partial M}{\partial t}, \quad \frac{\partial S}{\partial x} = \frac{\partial S}{\partial y} = 0.$$

Рассмотрим граничные условия на поверхности дна акватории при $z = -h(x,y)$, $(x,t) \in \Omega_h$ после обезразмеривания:

$$\frac{\mu_t}{E_H}\frac{\partial u}{\partial z}n_z - \frac{L}{H_c}\frac{Ki}{Eu}pn_x = -\frac{b_f u_c}{H_c f_c}u\sqrt{u^2 + v^2 + \frac{(w_c w)^2}{u_c^2}},$$
$$\frac{\mu_t}{E_H}\frac{\partial v}{\partial z}n_z - \frac{L}{H_c}\frac{Ki}{Eu}pn_y = -\frac{b_f u_c}{H_c f_c}v\sqrt{u^2 + v^2 + \frac{(w_c w)^2}{u_c^2}},$$
$$w|_{z=-h(x,y)} = -u|_{z=-h(x,y)}\frac{\partial h}{\partial x} - v|_{z=-h(x,y)}\frac{\partial h}{\partial y}, \qquad \mathbf{v}\cdot\mathbf{n} = 0,$$
$$T|_{z=-h} = T_h(x,y), \qquad S|_{z=-h} = S_h(x,y), \qquad (x,y) \in \Omega. \tag{23}$$

На участке границы входа Γ_{in} задается уровень поверхности воды (приливы – отливы) и скорость течения (13), тепловые потоки и поток массы соли равны нулю. В качестве вектора скорости на входе принимается скорость, найденная из безразмерных уравнений динамики воды, записанных для установившегося дрейфового течения:

$$\frac{\mu_t}{E_H}\frac{\partial^2 u_g}{\partial z^2} = -v_g, \qquad \frac{\mu_t}{E_H}\frac{\partial^2 v_g}{\partial z^2} = u_g, \qquad (x,y) \in \Gamma_{in},$$

с краевыми условиями:

$$\frac{\mu_t}{E_H}\frac{\partial u_g}{\partial z}\Big|_{z=\xi} = \frac{\rho_a C_d}{\rho_0 u_c f_c H_c} W_x |\mathbf{W}|, \qquad \frac{\mu_t}{E_H}\frac{\partial v_g}{\partial z}\Big|_{z=\xi} = \frac{\rho_a C_d}{\rho_0 u_c f_c H_c} W_y |\mathbf{W}|, \qquad (x,y) \in \Gamma_{in},$$

$$u_g\big|_{z=-\infty} \neq \infty, \qquad v_g\big|_{z=-\infty} \neq \infty. \tag{24}$$

Для определения уровня поверхности воды участке границы входа Γ_{in} получим выражение

$$\Psi|_{\Gamma_{in}} = \int_{-h}^{0} \rho\, dz + \xi_0, \;\; \rho = \rho(T,S), \;\; (x,y) \in \Gamma_{in}. \tag{25}$$

На свободной границе Γ_{out} считаем, что потоки либо отсутствуют, либо задан квадратичный закон трения, кроме того

$$\frac{\partial \Psi}{\partial n} = 0, \quad (x,t) \in \Gamma_{out}. \tag{26}$$

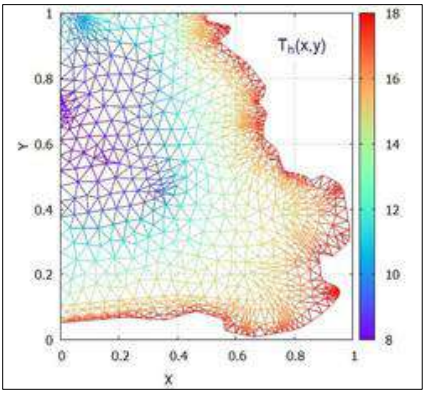


Рис. 4. Распределение поля температур на дне акватории
Fig. 4. Distribution of the temperature field at the bottom of the water area

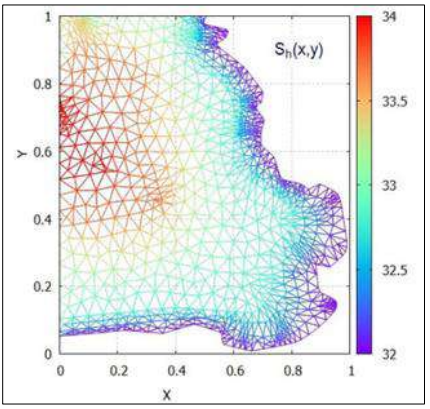


Рис. 5. Распределение диффузии солености на дне акватории
Fig. 5. Distribution of salinity diffusion at the bottom of the water area

Начальное распределение поверхности уровня воды, температуры, солености и вектора скорости задано в (13). Для определения значений температуры и солености на дне акватории в поставленной задаче в качестве краевых условий использовались данные измерений соответствующих величин на разных глубинах Тихого океана. Графики поверхностей, описывающих распределение температуры и соли в летний период на дне акватории, представлены на рис.4, 5.

5. Вычислительный эксперимент

Поставленная задача решается методом конечных элементов с дискретизацией по времени всех уравнений. Применение неявной схемы и процедуры Ньютона для линеаризации нелинейных слагаемых в уравнениях движения обеспечивает сходимость итерационного процесса на достаточно больших временных шагах.

В качестве характерных значений основных параметров и величин примем следующие значения:

$$t_c = 10^4\ [\text{с}];\ L_x = 2000\ [\text{м}];\ L_y = 3150\ [\text{м}];\ L = 3150\ [\text{м}];\ H = 10\ [\text{м}];$$
$$u_c = 1\ [\text{м/с}];\ p_c = 1\ [\text{Па}];\ \rho_0 = 1024.12\ [\text{кг/м}^3];\ f_c = 10^5\ [\text{с}^{-1}];$$
$$K = 10^3[\text{м}^2/\text{с}];\ \nu_t = 10^3[\text{м}^2/\text{с}];\ K_T = 10^{-1}[\text{м}^2/\text{с}].$$

Расчет модели проводится на высокоуровневом мультифизическом программном обеспечении FreeFem++ с конечными элементами на геометрии бухты Аякс Японского моря. Общее время наблюдения 2.7 ч. Сетка по всему объему содержит 17620 конечных элементов. Построенная сетка является неравномерной. В местах далеких от прибрежной линии наибольший объемный размер одного элемента составляет 157.5×157.5× 1 м^3 ближе к линии берега наименьший размер объема элемента не превосходит значений 0.945×0.945×1 м^3, шаг по времени соответствует 0.27 ч.

Для решения гиперболического уравнения системы используется параболическая регуляризация с применением обратного метода Эйлера второго порядка для аппроксимации производной по времени. При решении сингулярной задачи (20), (25), (26) с малым параметром регуляризации $\varepsilon > 0$, применяется адаптация двумерной сетки вокруг численного решения Ψ, обновляемая для каждого временного слоя. Управляющими параметрами сетки, определяемые пользователем, являются следующие параметры:

- уровень ошибки интерполяции внутри геометрии $err = 10^{-5}$,
- уровень ошибки интерполяции на границе $errg = 10^{-2}$,
- изотропность метрики $iso = true$,
- максимальное количество вершин, разрешенных в генераторе сетки $nvbox = 10^7$,
- максимальный линейный размер конечного элемента $hmax = 10^{-2}$.

Минимальный линейный размер элемента адаптированной сетки для всех итераций составил $\text{hmin} = 0.000308126$, количество конечных элементов принадлежит диапазону значений от 16811 до 16869. При данных параметрах сетки, параметр регуляризации задачи выбран $\varepsilon = 10^{-4}$.

Основными причинами, определяющими скоростную структуру водного потока, являются дрейф водных масс под действием ветра и приливно-отливные колебания уровня воды. При выборе гидрологического сценария предполагалось, что сила ветра над уровнем моря 10 м. равна $\mathbf{W} = \{7; 9\}$, $|\mathbf{W}| = 11.4$ м/с. Время приливного цикла задано выражениями $u = u_g + 0.03t$, $v = v_g + 0.03t$, $\xi = -0.3/H + 0.8\,t/H$, $t \in (0,1)$, где $\{u_g;\ v_g\}$ – дрейфовое течение.

На рисунках 6–9 представлены векторные поля скоростей в акватории бухты Аякс в начальный момент времени и через 2.7 часа. На рис.6, 7 указано направление вектора ветра **W.** На рисунках 6-13 представлены графики безразмерных характеристик на безразмерной

геометрии для действительных значений времени. При дальнейшем увеличении времени для данного гидрологического сценария гидродинамическая картина не меняется.

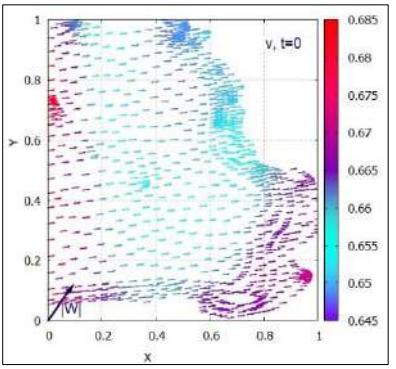


Рис. 6. Поле скоростей на поверхности бухты в начальный момент времен
Fig. 6. Velocity field on the surface of the bay at the initial moment of time

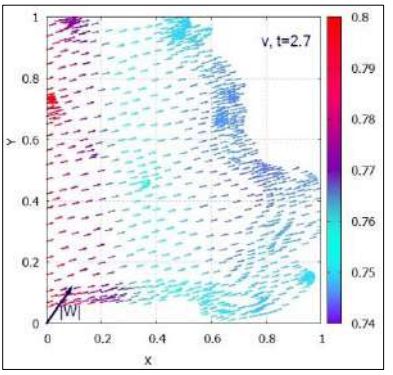


Рис. 7. Поле скоростей на поверхности бухты в моменты времени t=2.7 ч. от начала расчета
Fig. 7. Velocity field on the surface of the bay at time t=2.7 hours from the beginning of the calculation.

Сравним осредненные по вертикальному направлению значения модуля вектора-скорости $mod(v) = \sqrt{\bar{u}^2 + \bar{v}^2}$ в начальный момент времени и через 2.7 часа после начала расчета. Полученные поверхности модуля скорости представлены на рис. 8, 9.

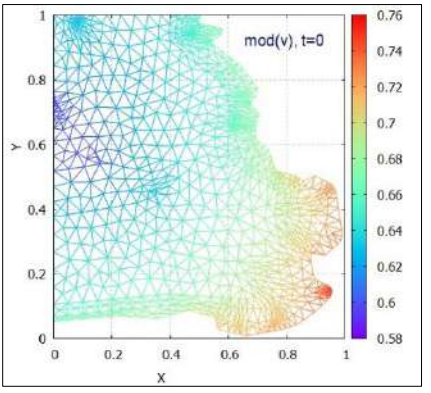


Рис. 8. Распределение осредненного по вертикали модуля вектора скорости в начальный момент времени
Fig. 8. Distribution of the vertically averaged modulus of the velocity vector at the initial moment of time

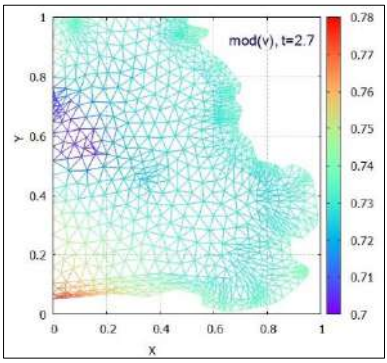


Рис. 9. Распределение осредненного по вертикали модуля вектора скорости в момент времени t=2.7 ч
Fig. 9. Distribution of the vertically averaged modulus of the velocity vector at time t=2.7 h

На пространстве непрерывных функций вычислим норму функции коэффициента вертикальной турбулентности и норму вертикальной компоненты вектора скорости по формулам $K(t) = \max_{(x,y,z)\in Q} |K(t)|$, $w(t) = \max_{(x,y,z)\in Q} |w(t)|$.

Графики функций w(t), K(t) представлены на рис. 10, 11 в разные моменты времени.

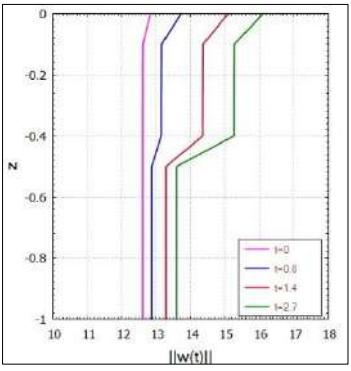


Рис. 10. Графики значений функции $w(t)$ в моменты времени t=0 ч., 0.83 ч., 1.4 ч., 2.7 ч. от начала расчета
Fig. 10. Graphs of the values of the function w(t) at the time points t=0 h., 0.83 h., 1.4 h., 2.7 h. from the beginning of the calculation

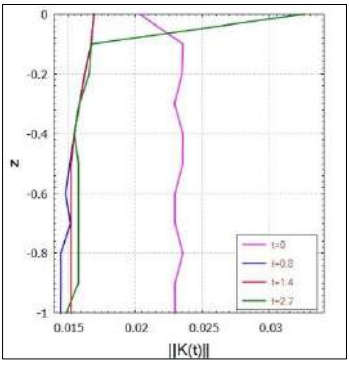


Рис.11. Графики значений функции $K(t)$ в моменты времени t=0 ч., 0.83 ч., 1.4 ч., 2.7 ч. от начала расчета
Fig. 11. Graphs of the values of the function K(t) at the time points t=0 h., 0.83 h., 1.4 h., 2.7 h. from the beginning of the calculation

Безразмерная функция свободной поверхности $\xi(x, y, t)$ в начальный момент времени и через 2.7 часа представлена на рисунках 12,13.

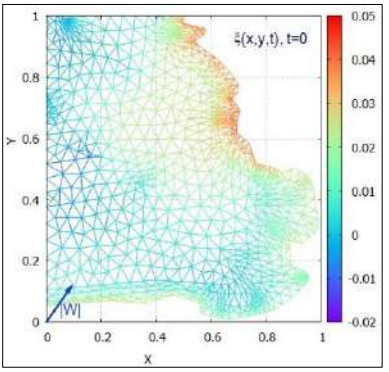


Рис. 12. График значений функции $\xi(x, y, t)$ в начальный момент времени
Fig. 12. Graph of the values of the function ξ(x,y,t) at the initial moment of time

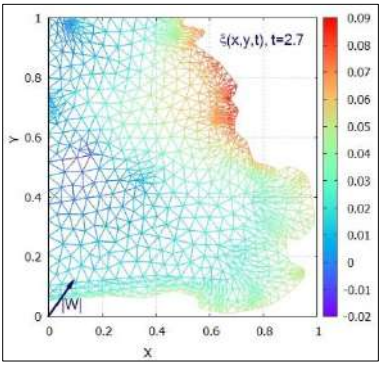


Рис.13. График значений функции $\xi(x, y, t)$ в момент времени t=2.7 ч. от начала расчета
Fig. 13. Graph of the values of the function ξ(x,y,t) at the time t=2.7 hours from the start of the calculation

6. Заключение

В работе представлен результат численного моделирования гидродинамического сценария в акватории залива Петра Великого Японского моря. Основу исследования составляет трехмерная модель осредненных по времени уравнений Навье-Стокса в условиях приближения гидростатики и уравнений диффузии, учитывающих адвективный перенос температуры и соли. Важным фактором негативного влияния воздействия на водную среду при производстве гидростроительных работ является загрязнение водной среды вследствие выхода во взвесь низко дисперсных фракций. В результате происходит замутнение воды и, следовательно, ослабление процессов нормального развития зообентоса.

Численное моделирование развивающихся гидродинамических процессов позволит не только оценить ущерб, нанесенный ихтиофауне в процессе производства работ в морской акватории, но и спрогнозировать оптимальный режим требуемых работ с учетом ветровых погодных условий, сезонных приливов и отливов.

Список литературы

[1] Кудрявцева О.В., Ледащева Т.Н., Пинаев В.Е. Методика и практика оценки воздействия на окружающую среду. Проектная документация. Учебное пособие. 2-е издание. М., Экономический факультет МГУ имени М. В. Ломоносова, 2018 г., 160 стр. / Kudryavtseva O.V., Ledashcheva T.N.,

Pinaev V.E. Methodology and practice of environmental impact assessment. Project documentation. Tutorial. 2nd edition. M., Faculty of Economics of Moscow State University, 2018, 160 p. (in Russian).

[2] Водный кодекс Российской Федерации (с изменениями на 1 мая 2022 года). Электронный фонд правовых и нормативно-технических документов / Water Code of the Russian Federation (as amended as of May 1, 2022). Electronic fund of legal and normative-technical documents, URL: https://docs.cntd.ru/document/901982862 (in Russian).

[3] Горбачев С.А. Вопросы оценки ущерба водным биоресурсам. Труды Кольского научного центра РАН, вып. 4, 2012 г., стр. 46–62 / Gorbachev S. A. Questions of assessment of water bioresources damage. Transactions of the Kola Science Centre, issue 4, 2012, pp. 46-62 (in Russian).

[4] Богдановский А.А., Кочергин И. Е. Параметризация характеристик перемешивания для типичных условий северо-восточного шельфа Сахалина. Труды ДВНИГМИ, тематический выпуск «Гидрометеорологические процессы на шельфе: оценка воздействия на окружающую среду», 1998 г., стр. 89-102 / Bogdanovskiy A.A., Kochergin I.E. Parameterization of mixing characteristics for typical conditions of the northeastern Sakhalin shelf. Proceedings of the FERHMI, thematic issue «Hydrometeorological processes on the shelf: environmental impact assessment», 1998, pp. 89-102 (in Russian).

[5] Котеров В.Н., Юрезанская Ю.С. Моделирование переноса взвешенных веществ на океаническом шельфе. Эффективная гидравлическая крупность полидисперсной взвеси. Журнал вычислительной математики и математической физики, том 49, вып. 7, 2009 г., стр. 1306–1319 / Koterov V.N., Yurezanskaya Yu.S. Simulation of suspended substance dispersion on the ocean shelf: Effective hydraulic coarseness of polydisperse suspension. Computational Mathematics and Mathematical Physics, vol. 49, issue 7, 2009, pp. 1245-1256.

[6] Сухинов А.И., Чистяков А.Е., Проценко Е.А. Математическое моделирование транспорта насосов в прибрежных водных системах на многопроцессорной вычислительной системе. Вычислительные методы и программирование, том 15, вып. 4, 2014 г., стр. 610–620 / Sukhinov A.I., Chistyakov A.E., Protsenko E.A. Sediment Transport Mathematical Modeling in a Coastal Zone Using Multiprocessor Computing Systems. Numerical Methods and Programming, vol. 15, issue 4, 2014, pp. 610-620 (in Russian).

[7] Амосова Е.В., Кикелин Д.С. Математическое моделирование распространения зон загрязнения взвесью в морской среде. Подводные исследования и робототехника, вып. 4(38), 2021 г., стр. 72-79 / Amosova E.V., Kikelin D.S. Mathematical modeling of the suspended pollution zones spread in the marine environment. Underwater Investigations and Robotics, issue 4(38), 2021, pp. 72-79 (in Russian).

[8] Наумов В.А. Математическое моделирование распространения взвешенных примесей от точечного источника и их осаждения в водостоке. Известия КГТУ, вып. 44, 2017 г., стр. 46–58 / Naumov V.A. Mathematical modeling of distribution of suspended impurities from a point source and its deposition in the watercourse. KSTU News, issue 44, 2017, pp. 46-58 (in Russian).

[9] Pinho J. L.S., Antunes do Carmo J.S., Vieira J.M.P. Mathematical modelling of oil spills in the Atlantic Iberian coastal waters. WIT Transactions on Ecology and the Environment, vol. 68, 2004. pp.337-347.

[10] Li D., Tang X. et al. Mathematical Modeling of Marine Oil Spills in the Luanjiakou District, near the Port of Yantai. Discrete Dynamics in Nature and Society, 2018, article ID 2736102, 22 p.

[11] Бухтеев В.Г., Доронин М.М. и др. Динамика океана. Л., Гидрометеоиздат, 1980 г., 303 стр. / Bukhteev V.G., Doronin M.M. et al. Dynamics of the ocean. L., Gidrometeoizdat, 1980, 303 p. (in Russian).

Информация об авторах / Information about authors

Елена Владимировна АМОСОВА – кандидат физико-математических наук, профессор департамента математического и компьютерного моделирования ИМКТ ДВФУ, научный сотрудник ИПМ ДВО РАН. Сфера научных интересов: дифференциальные уравнения, механика жидкости и газа, вычислительная гидродинамика.

Elena Vladimirovna AMOSOVA – Candidate of Physical and Mathematical Sciences, Professor of the Department of Mathematical and Computer Modeling of the IMCT FEFU, Researcher, IPM FEB RAS. Research interests: differential equations, fluid and gas mechanics, computational fluid dynamics.

Кирилл Сергеевич КУЗНЕЦОВ – ассистент департамента математического и компьютерного моделирования ИМКТ ДВФУ, аспирант второго года обучения департамента

математического и компьютерного моделирования ИМКТ ДВФУ, лаборант-исследователь ИПМ ДВО РАН. Его научные интересы включают машинное обучение, численные методы, искусственный интеллект.

Kirill Sergeevich KUZNETSOV Assistant of the Department of Mathematical and Computer Modeling, FEFU IMCT, second year postgraduate student of the Department of Mathematical and Computer Modeling, FEFU IMCT, laboratory researcher at IPM FEB RAS. His research interests include machine learning, numerical methods, artificial intelligence.

Виталий Сергеевич ЛЕМЕШЕВ является аспирантом первого года обучения отделения машиностроения, морской техники и транспорта инженерного департамента ПИ ДВФУ. Научные интересы включают численные методы, механика.

Vitaly Sergeevich LEMESHEV is a first-year postgraduate student of the Department of Mechanical Engineering, Marine Engineering and Transport of the Engineering Department of the PI FEFU. Scientific interests include numerical methods, mechanics.

DOI: 10.15514/ISPRAS-2022-34(5)-17

О решении одной задачи мелкой воды методом центральных разностей и коррекцией FCT

И.И. Потапов, ORCID: 0000-0002-3323-2727 <potapov2i@gmail.com>
П.С. Тимош, ORCID: 0000-0002-3132-2227 <pavel.timosh@yandex.ru>

Вычислительный центр ДВО РАН (ХФИЦ ДВО РАН),
680000, Россия, г. Хабаровск, ул. Ким Ю Чена, д. 65

Аннотация. В работе предложен алгоритм решения задачи мелкой воды, реализованный на основе центрально-разностной по пространству и явной одношаговой по времени схемы, устойчивость которой достигается методом FCT. Показано, что при решении задачи о распаде разрыва цилиндрического столба жидкости в мелком бассейне, предложенный метод является устойчивым, сравним по точности с методом Мак–Кормака превосходя его по производительности.

Ключевые слова: мелкая вода; центрально–разностная схема; FCT; распад разрыва

Для цитирования: Потапов И.И., Тимош П.С. О решении одной задачи мелкой воды методом центральных разностей и коррекцией FCT. Труды ИСП РАН, том 34, вып. 5, 2022 г., стр. 243-250. DOI: 10.15514/ISPRAS-2022-34(5)-17

Solving the shallow water problem by central differences and FCT correction

I.I. Potapov, ORCID: 0000-0002-3323-2727 <potapov2i@gmail.com>
P.S. Timosh, ORCID: 0000-0002-3132-2227 <pavel.timosh@yandex.ru>

Computing Center of the Far Eastern Branch of the Russian Academy of Sciences,
65, Kim Yu Chen st., Khabarovsk, 680000, Russia

Abstract. The paper proposes an algorithm based on central differences and FCT correction for solving the shallow water problem. The results of numerical testing were compared with known data. The comparison showed that the proposed algorithm has similar accuracy with other methods. A comparison of the speed of the proposed algorithm and a similar one based on the McCormack method is given. The conclusion is made about the superiority in speed over the McCormack method with the same accuracy.

Keywords: shallow water; central difference scheme; FCT; discontinuity decay problem

For citation: Potapov I.I., Timosh P.S. Solving the shallow water problem by central differences and FCT correction. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 5, 2022. pp. 243-250 (in Russian). DOI: 10.15514/ISPRAS-2022-34(5)-17

1. Введение

Ведение хозяйственной деятельности на пойменных территориях часто влияет на гидродинамические характеристики реки. Поэтому верные и точные количественные оценки влияния хозяйственной деятельности в руслах и поймах рек важны для научного обоснования предлагаемых проектных решений. Математическое моделирование гидравлики речных процессов часто происходит в рамках теории мелкой воды. В данной работе используется плановая постановка задачи мелкой воды.

Из-за гиперболических свойств уравнений мелкой воды большой интерес вызывают методы, способные давать монотонное и консервативное решение задачи Римана при распаде гидравлического разрыва. За последние сто лет было предложено множество способов численного решения задачи Римана с порядком точности выше первого. Например, такие как схемы Лакса [1] и Лакса—Вендоффа [2], Мак-Кормака [3], введение искусственной вязкости [4, 5].

В работе Годунова [6] было показано, что невозможно получить линейный метод решения с порядком точности выше первого. Дальнейшие исследования пошли в направлении создания нелинейных методов решения. Были предложены такие методы, как FCT [7, 8], TVD [9, 10].

В данной работе метод, описанный в [11], расширен для решения задачи мелкой воды. Произведено сравнение метода по производительности с другими применяемыми методами.

2. Постановка задачи

В области $\Omega(0 \le x \le l,\ 0 \le y \le m)$ с границами Г, решается задача мелкой воды.

Математическая модель задачи определяется системой дифференциальных уравнений [12]:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = \Lambda, \tag{1}$$

$$\mathrm{Q} = \begin{bmatrix} h \\ q_x \\ q_y \end{bmatrix}, \mathrm{F} = \begin{bmatrix} q_x \\ \frac{q_x^2}{h} + \frac{1}{2} g(h^2 + h\zeta) \\ \frac{q_x q_y}{h} \end{bmatrix}, \mathrm{G} = \begin{bmatrix} q_y \\ \frac{q_x q_y}{h} \\ \frac{q_y^2}{h} + \frac{1}{2} g(h^2 + h\zeta) \end{bmatrix}, \tag{2}$$

$$\Lambda = \begin{bmatrix} 0 \\ -\lambda \sqrt{u^2 + v^2} u \\ -\lambda \sqrt{u^2 + v^2} v \end{bmatrix}. \tag{3}$$

где Q — вектор консервативных переменных, F и G — векторы потоков вдоль осей x и y, Λ — вектор гидравлического сопротивления; q_x, q_y — компоненты вектора расходов воды (м^2/с), h — глубина (м), ζ — уровень дна канала (м), g — ускорение свободного падения (м/с^2), λ — коэффициент гидравлического трения; $\mathrm{u} = \frac{q_x}{h}$ и $\mathrm{v} = \frac{q_y}{h}$ (м/с).

На границе Г области Ω заданы условия:

$$\frac{\partial \mathrm{h}}{\partial \mathrm{x}} \mathrm{n_x} + \frac{\partial \mathrm{h}}{\partial \mathrm{y}} \mathrm{n_y} = \mathrm{h(x, y, t)}. \tag{4}$$

$\mathrm{n_x}$ и $\mathrm{n_y}$ — компоненты вектора нормали на границе Г.

В начальный момент времени заданы распределение компонент вектора расхода и глубина:

$$q_x(x, y, 0) = q_{x_0}(x, y),\ q_y(x, y, 0) = q_{y_0}(x, y),\ \mathrm{h}(x, y, 0) = h_0(x, y),\ \mathrm{x, y} \in \Omega. \tag{5}$$

3. Метод решения

Для решения задачи (1)–(5) предлагается использовать явную схему конечных разностей: разность вперед по времени и центральная разность по пространству. Монотонность решения центрально-разностной схемы обеспечивается методом FCT.

При решении задачи (1)–(5) для расчетной области Ω вводится дискретный аналог: $x_i = \Delta x \cdot i,\ \mathrm{i} = \overline{0 \dots L},\ y_j = \Delta y \cdot j,\ \mathrm{j} = \overline{0 \dots M},\ t_n = \Delta t \cdot n,\ \mathrm{n} = \overline{0 \dots N}$. Здесь $\Delta x = \frac{1}{L}$, $\Delta y = \frac{1}{M}$, — шаги сетки по пространству, Δt — по времени; l, m — длинна и ширина области, L, M — количество узлов по осям x и y, N — количество узлов по времени.

Ниже приведен алгоритм для решения задачи (1) – (5).

Шаг 1. Вычисление предварительного решения

$$Q^{+}_{i,j} = Q^{n}_{i,j} - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(F^{n}_{i+1,j} - F^{n}_{i-1,j}\right) - \frac{1}{2}\frac{\Delta t}{\Delta y}\left(G^{n}_{i,j+1} - G^{n}_{i,j-1}\right) + \Delta t \Lambda^{n}_{i,j}$$

Шаг 2. Вычисление диффузионных

$$f^{d}_{i,j} = \nu_{i,j}\left(Q^{n}_{i+1,j} - Q^{n}_{i,j}\right),$$
$$g^{d}_{i,j} = \mu_{i,j}\left(Q^{n}_{i,j+1} - Q^{n}_{i,j}\right),$$

и антидиффузионных потоков

$$f^{ad}_{i,j} = \chi_{i,j}\left(Q^{+}_{i+1,j} - Q^{+}_{i,j}\right),$$
$$g^{ad}_{i,j} = \kappa_{i,j}\left(Q^{+}_{i,j+1} - Q^{+}_{i,j}\right),$$

где

$$\nu_{i,j} = \eta_0 + \eta_1\,\alpha^2_{i,j},\ \ \mu_{i,j} = \eta_0 + \eta_1\,\epsilon^2_{i,j};$$
$$\chi_{i,j} = \eta_0 + \eta_2\,\alpha^2_{i,j},\ \ \kappa_{i,j} = \eta_0 + \eta_2\,\epsilon^2_{i,j};$$
$$\eta_0 = \frac{1}{6},\ \ \eta_1 = \frac{1}{3},\ \ \eta_2 = \frac{1}{6};$$
$$\alpha_{i,j} = \frac{u^{n}_{i+1,j} + u^{n}_{i,j}}{2}\,\frac{\Delta t}{\Delta x},\ \ \epsilon_{i,j} = \frac{v^{n}_{i,j+1} + v^{n}_{i,j}}{2}\,\frac{\Delta t}{\Delta y}.$$

Шаг 3. Диффузия решения

$$Q^{*}_{i,j} = Q^{+}_{i,j} + f^{d}_{i,j} - f^{d}_{i-1,j} + g^{d}_{i,j} - g^{d}_{i,j-1}.$$

Шаг 4. Расчет первых разностей

$$\Delta_x Q^{*}_{i,j} = Q^{*}_{i,j} - Q^{*}_{i-1,j},$$
$$\Delta_y Q^{*}_{i,j} = Q^{*}_{i,j} - Q^{*}_{i,j-1}.$$

Шаг 5. Ограничение антидиффузионных потоков

$$f^{cad}_{j} = S^{f}_{i,j}\max\left\{0, \min\left[S^{f}_{i,j}\Delta_x Q^{*}_{i-1,j}, \left|f^{ad}_{i,j}\right|, S^{f}_{i,j}\Delta_x Q^{*}_{i+1,j}\right]\right\},$$
$$g^{cad}_{j} = S^{g}_{i,j}\max\left\{0, \min\left[S^{g}_{i,j}\Delta_y Q^{*}_{i,j-1}, g^{ad}_{i,j}\right|, S^{g}_{i,j}\Delta_y Q^{*}_{i,j+1}\right]\right\};$$
$$S^{f}_{i,j} = \operatorname{sign} f^{ad}_{i,j},\ \ S^{g}_{i,j} = \operatorname{sign} g^{ad}_{i,j}.$$

Шаг 6. Антидиффузионная коррекция решения

$$Q^{n+1}_{i,j} = Q^{*}_{i,j} - \left(f^{cad}_{i,j} - f^{cad}_{i-1,j} + g^{cad}_{i,j} - g^{cad}_{i,j-1}\right).$$

Устойчивость решения определяется следующим условием:

$$\Delta t \le C\frac{\min(\Delta x, \Delta y)}{\sqrt{u^2_{max} + v^2_{max} + g^2 h^2_{max}}}.$$

Здесь C – число Куранта.

4. Тестирование и результаты

Тестирование предложенного алгоритма выполнено на классических тестовых задачах, приведенных в работах [13] и [16].

4.1 Распад разрыва над горизонтальным дном

Первым тестом является решение одномерной задачи о распаде разрыва над ровным дном [13]. Рассматривается плоское одномерное течение жидкости в канале длины $l = 240$ м с плоским дном $\zeta = \text{const}$. В начальный момент в центре области задается разрыв уровня воды, разделяющий два однородных состояния с высотой $h = h_L$ слева и $h = h_R$ справа от разрыва;

справа и слева от разрыва жидкость неподвижна: $u_L = u_R = 0$, $h_L = 10$ м, $h_R = 1$ м. Параметры сетки: $l = 240$ м и $m = 50$ м; $\Delta x = 0{,}25$ м и $\Delta y = 0{,}25$ м, $\Delta t = 0{,}001$ с.

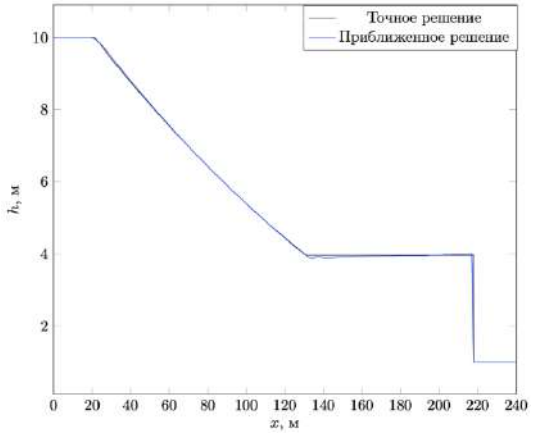


Рис. 1. Уровень глубины в задаче о распаде начального разрыва при $t = 10$ с.
Fig. 1. Depth level in the initial discontinuity decay problem at $t = 10$ s

Результаты в виде графика глубины приведены для момента времени $t = 10$ с на рис. 1.

4.2 Задача о распаде цилиндрического разрыва

Второй тест выполнен на задаче об изменение уровня свободной поверхности при разрушении цилиндрического столба несжимаемой невязкой жидкости в мелком бассейне [13–15]. На рис. 2 представлена расчетная область Ω.

На границе Γ области Ω заданы условия свободного протекания:

$$\frac{\partial h}{\partial x} n_x + \frac{\partial h}{\partial y} n_y = 0.$$

В начальный момент времени заданы распределение компонент вектора расхода и глубина:

$$q_x(x,y,0) = 0,\ q_y(x,y,0) = 0,\ h(x,y,0) = \begin{cases} h_C, \text{ при } x^2 + y^2 \le r^2 \\ h_0, \text{ при } x^2 + y^2 > r^2 \end{cases}, x, y \in \Omega.$$

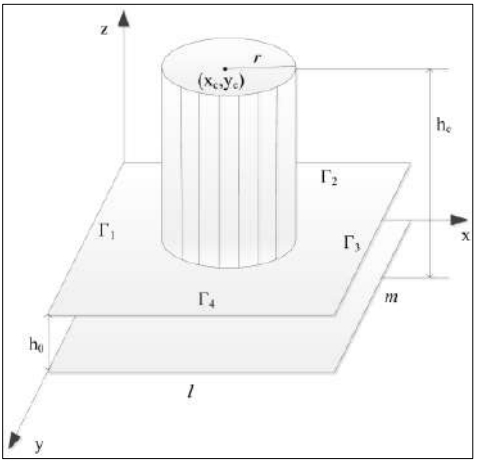


Рис. 2. Схема расчетной области задачи о распаде цилиндрического разрыва.
Fig. 2. Scheme of the computational domain of the cylindrical discontinuity decay problem

Тестирование проведено при следующих параметрах: $l = 50$ м и $m = 50$ м; $\Delta x = 0{,}25$ м и $\Delta y = 0{,}25$ м, $\Delta t = 0{,}001$ с; $r = 10$ м — радиус цилиндрического столба жидкости с

центром в точке (x_c, y_c); $h_c = 2$ м — глубина в начальный момент времени в пределах столба, и $h_0 = 0{,}5$ м — глубина для остальной части области.

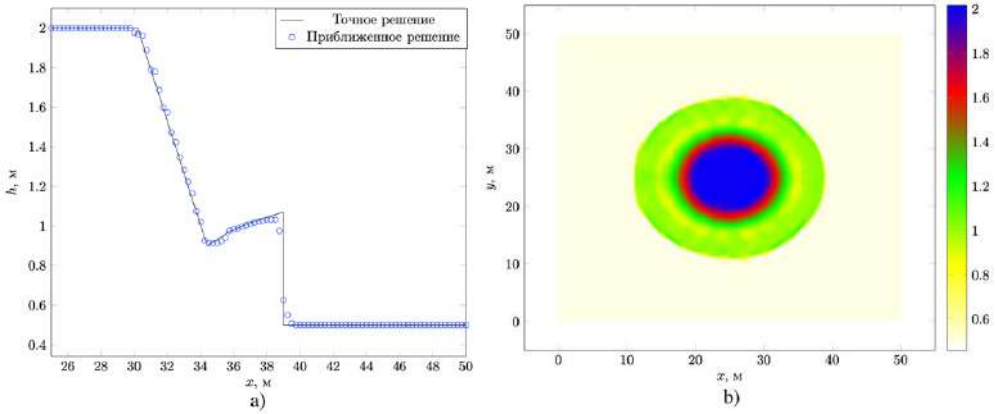


Рис. 3. Тест задачи о распаде цилиндрического разрыва при $t = 1$ с.
Fig. 3. Test of cylindrical discontinuity decay problem at $t = 1$ s.

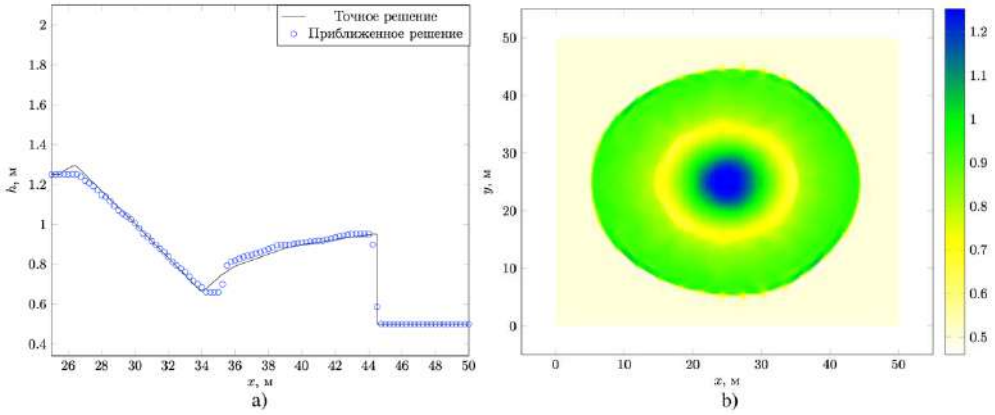


Рис. 4. Тест задачи о распаде цилиндрического разрыва при $t = 2{,}5$ с.
Fig. 4. Test of cylindrical discontinuity decay problem at $t = 2{,}5$ s

Результатами расчетов являются значения функции h. На рис. 3 приведены результаты для 1 с: график профиля функции h от центра до края области (3a) и проекция распределения глубины на плоскость Oxy (3b). Результаты расчетов показывают, что алгоритм плохо улавливает перепады при выбранных шагах по пространству. Это особенно заметно на графике профиля глубины для 2,5 с (рис. 4), что можно связать с особенностями метода антидиффузионной коррекцией потока.

Проекции глубины демонстрируют равномерное, симметричное распространение волны по всем направлениям. Полученные результаты подтверждают применимость алгоритма к решению задачи мелкой воды в областях, содержащих распад разрыва.

4.3 Течения с бугром на дне

В данном тесте выбраны следующие параметры: $l = 25$ м и $m = 15$ м; $\Delta x = 0{,}25$ м и $\Delta y = 0{,}25$ м, $\Delta t = 0{,}001$ с. Рельеф дна задан по формуле, приведенной в [13]:

$$\zeta(x) = \begin{cases} 0{,}2 - 0{,}05(x - 10)^2, & l/3 < x < l/2; \\ 0, \text{ иначе.} \end{cases}$$

Начальные условия: $u = 0$, $\zeta + h = 2$ м. При $x = 0$ расход равен $q_x = 4{,}42$ м^2/с, а при $x = 25$ м глубина $h = 2$ м.

Точное решение $h = h(x)$ можно найти с помощью соотношения [13]:

$$h^3 + \left(\zeta - \frac{q_{in}^2}{2gh_{out}^2} - h_{out}\right)h^2 + \frac{q_{in}^2}{2g} = 0.$$

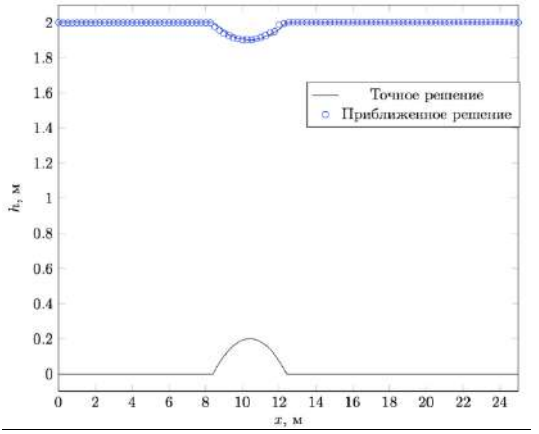


Рис. 5. Уровень глубины в задаче о течении с бугром на дне (докритический режим) при $t = 10$ с.
Fig. 5. Depth level in the problem of a flow with a hillock at the bottom (subcritical regime) at $t = 10$ s

Результаты приведены для момента времени $t = 10$ с на рис. 5. При времени равном 10 секундам течение является установившимся. При этом рассчитанные значения совпадают с точным решением.

5. Заключение

В работе предложен метод решения плановой задачи мелкой воды. Метод реализован на основе центрально-разностной по пространству и явной одношаговой по времени схемы, устойчивость которой достигается использованием метода антидиффузионной коррекции потоков FCT.

Работа метода продемонстрирована на примере решения классических тестовых задачах, приведенных в работах [13] и [16]. Результаты моделирования сравнивались с известными данными [13-16]. Сравнение показало, что предложенный метод имеет сравнимую точность решения.

Приведенный алгоритм был реализован на языке C# и платформе .Net 6. Для сравнения скорости вычислений также был реализован метод Мак-Кормака с использованием FCT. Измерения показали, что скорость вычислений предложенного алгоритма на 40 процентов выше.

Список литературы / References

[1] Lax P.D. Weak Solutions of Nonlinear Hyperbolic Equations and their Numerical Computation. Communications on Pure and Applied Mathematics, vol. 7, 1954, pp. 159-193.
[2] Lax P.D., Wendroff B. Systems of Conservation Laws. Communications on Pure and Applied Mathematics, vol. 13, 1960, pp. 217-237.
[3] MacCormack R.W. The Effect of Viscosity in Hypervelocity Impact Cratering. AIAA Paper 69-354, 1969.
[4] Von Neumann J., Richtmyer R.D. A Method for the Numerical Calculation of Hydrodynamic Shocks. Journal of Applied Physics, vol. 21, issue 3, 1950, pp. 232-237.
[5] Lapidus A. A Detached Shock Calculation by Second-Order Finite Differences. Journal of Computational Physics, vol. 2, issue 2, 1967, pp. 154-177.

[6] Годунов С.К. Разностный метод численного расчета разрывных решений уравнений гидродинамики. Математический сборник. Новая серия, том 47(89), вып. 3, 1959 г., стр. 271-306 / Godunov S.K. Difference method for the numerical computation of discontinuous solutions of fluid dynamics equations. Matematicheskii Sbornik. Novaya Seriya, vol. 47(89), issue 3, 1959, pp. 271-306 (in Russian).
[7] Boris J.P., Book D.L. Flux-Corrected Transport. I. SHASTA, A Fluid Transport Algorithm that Works, J. Comp. Physics, 11, p. 38, 1973.
[8] Zalesak S.T. Fully multidimensional flux-corrected transport algorithms for fluids. Journal of Computational Physics, vol. 31, issue 3, 1979, pp. 335-362.
[9] Harten A. High Resolution Schemes for Hyperbolic Conservation Laws. Journal of Computational Physics, vol. 49, issue 3, 1983, pp. 357-393.
[10] Chakravarthy R., Osher S. A New Class of High Resolution TVD Schemes for Hyperbolic Conservation Laws. AIAA Paper 85-0363, 1985.
[11] Потапов И.И., Тимош П.С. Об использовании Центрально-разностной схемы для решения задачи газовой динамики. Информатика и системы управления, вып. 2(68), 2021 г., стр. 17–22 / Potapov I.I., Timosh P.S. On the use of the central difference scheme for solving the problem of gas dynamics. Information Science and Control Systems, issue 2(68), 2021, pp. 17-22 (in Russian).
[12] Картвелишвили Н.А. Неустановившиеся открытые потоки. Л., Гидрометеоиздат, 1968 г., 125 стр. / Kartvelishvili N.A. Unsettled open streams. L., Gidrometeoizdat, 1968, 125 p. (in Russian).
[13] Беликов В.В., Алексюк А.И. Модели мелкой воды в задачах речной гидродинамики. М., РАН, 2020 г., 346 стр. / [1]. Belikov V.V., Aleksyuk A.I. Shallow water models in problems of river hydrodynamics. M., RAS, 2020, 346 p. (in Russian).
[14] Amiri S.M., Talebbeydokhti N., Baghlani A. A two-dimensional well-balanced numerical model for shallow water equations. Scientia Iranica, vol. 20, issue 1, 2013, pp. 97–107.
[15] Canestrelli A., Dumbser M. et al. Well-balanced high-order centered schemes on unstructured meshes for shallow water equations with fixed and mobile bed. Advances in Water Resources, vol. 33, issue 3, 2010, pp. 291-303.
[16] Булатов О.В., Елизарова Т Г. Регуляризованные уравнения мелкой воды и эффективный метод численного моделирования течений в неглубоких водоемах. Журнал вычислительной математики и математической физики, том 51, вып. 1, 2011 г., pp. 170–184 / Bulatov O.V., Elizarova T.G. Regularized shallow water equations and an efficient method for numerical simulation of shallow water flows. Computational Mathematics and Mathematical Physics, vol. 51, issue1, 2011, pp. 160–173.

Информация об авторах / Information about authors

Игорь Иванович ПОТАПОВ – доктор физико-математических наук, профессор, заведующий лабораторией вычислительной механики. Сфера научных интересов: метод конечных элементов, метод контрольных объемов, метод частиц, решение систем алгебраических уравнений; механика сыпучих сред, механика гетерогенных сред, механика русловых процессов.

Igor Ivanovich POTAPOV – Doctor of Physics and Mathematics, Professor, Head of the Department of Computational mechanics. Research interests: finite element method, control volume method, particle method, solution of systems of algebraic equations; mechanics of granular media, mechanics of heterogeneous media, mechanics of channel processes.

Павел Сергеевич ТИМОШ — младший научный сотрудник, аспирант. Научные интересы: численные методы, механика русловых процессов.

Pavel Sergeevich TIMOSH is a research assistant. Research interests is numerical methods, mechanics of channel processes.