ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 32 Выпуск 6

ISSN Online 2220-6426
Volume 32 Issue 6

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2020

ИСП РАН

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

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

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

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

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

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

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

Труды ИСП РАН реферируются и/или индексируются в:

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

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

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

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

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

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

УДК004.45

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

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

С о д е р ж а н и е

Table of Contents

DOI: 10.15514/ISPRAS–2020–32(6)–1

Мониторинг и тестирование на основе многоуровневых спецификаций программ

[1,2,3] *А.К. Петренко, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>*
[1] *Д.В. Ефремов, ORCID: 0000-0002-9916-056X <efremov@ispras.ru>*
[1,2] *Е.В. Корныхин, ORCID: 0000-0001-9303-3132 <kornevgen@ispras.ru>*
[1,2,3] *В.В. Кулямин, ORCID: 0000-0003-3439-9534 <kulyamin@ispras.ru>*
[1,2,3,4] *А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*
[1] *И.В. Щепетков, ORCID: 0000-0002-5794-004X <shchepetkov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[3] *НИУ Высшая школа экономики,*
101978, Россия, г. Москва, ул. Мясницкая, д. 20
[4] *Московский физико-технический институт,*
141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. В исследованиях по формальным методам разработки и верификации программ много внимания уделяется вопросам построения многоуровневых спецификаций, отвечающих потребностям методологии пошаговой детализации и итеративной разработки. При верификации программ или их моделей наличие нескольких уровней спецификаций также упрощает доказательство свойств, поскольку, как правило, при добавлении нового уточняющего уровня удается переиспользовать те доказательства, которые были выполнены для более абстрактных уровней модели. При тестировании на основе формальных моделей и при мониторинге систем с целью проверки соответствия поведения системы требованиям, заданными формальной моделью, желательно пользоваться теми же моделями, которые использовались при формальной верификации. На практике такие модели в крупных программных системах являются многоуровневыми, однако опыта их использования как основы тестирования и мониторинга пока не было. В статье рассматриваются различные методы построения многоуровневых моделей, новые возможности, которые удается извлечь за счет комбинации функциональных спецификаций и спецификаций архитектуры реализации, ограничения, которые приходится учитывать при организации тестирования и мониторинга на основе многоуровневых моделей.

Ключевые слова: формальные модели программ; уточнение; модель архитектуры программы

Для цитирования: Петренко А.К., Ефремов Д.В., Корныхин Е.В., Кулямин В.В., Хорошилов А.В., Щепетков И.В. Мониторинг и тестирование на основе многоуровневых спецификаций программ. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 7-18. DOI: 10.15514/ISPRAS–2020–32(6)–1

Благодарности. Работа поддержана грантом РФФИ № 20-01-00568.

Monitoring and Testing Based on Multi-Level Program Specifications

[1,2,3] *A.K. Petrenko, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>*
[1] *D.V. Efremov, ORCID: 0000-0002-9916-056X <efremov@ispras.ru>*
[1,2] *E.V. Kornykhin, ORCID: 0000-0001-9303-3132 <kornevgen@ispras.ru>*
[1,2,3] *V.V. Kuliamin, ORCID: 0000-0003-3439-9534 <kulyamin@ispras.ru>*
[1,2,3,4] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*
[1] *I.V. Shchepetkov, ORCID: 0000-0002-5794-004X <shchepetkov@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia,
[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[3] *National Research University, Higher School of Economics*
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
[4] *Moscow Institute of Physics and Technology (MIPT)*
141700, Russia, Moscow region, Dolgoprudny, Campus per., 9

Abstract. Research on formal methods of software development and verification focuses on building specifications using incremental and iterative development methodologies. The presence of several levels of specifications simplifies proving of properties, since it is possible to reuse the proofs that were performed for more abstract layers of the model. It is desirable to use the same models that were used for formal verification also in testing of real systems for compliance with the requirements set by these models. In practice, large software systems are described by multi-level models. There was no experience of using such models as the basis for testing and monitoring. The paper discusses various methods for developing multi-level models, new opportunities that can be obtained through a combination of functional specifications and implementation-level refinements, limitations that must be considered during testing and monitoring of real systems for compliance with multi-level models.

Keywords: software formal models; refinement; software architecture models

For citation: Petrenko A.K., Efremov D.V., Kornykhin E.V., Kuliamin V.V., Khoroshilov A.V., Shchepetkov I.V. Monitoring and testing based on multi-level program specifications. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 7-18 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–1.

Acknowledgments. This work was supported by the RFBR grant No. 20-01-00568.

1. Введение. Источники потребности в многоуровневых моделях

В последние годы все большее внимание уделяется методической и инструментальной поддержке жизненного цикла разработки доверенного программного обеспечения. В настоящее время имеющаяся цепочка покрывает значительное число процессов разработки и верификации ПО, но есть еще и слабо поддержанные виды работ. Одним из таких видов является тестирование на соответствие требованиям к системе. Причиной такого отставания является то, что в отличие от статического анализа или динамического фаззинга значимый эффект от тестирования на соответствие требованиям возможен только тогда, когда эти требования описаны достаточно полно и формализованы. Хотя инструменты генерации тестов на основе формальных спецификаций или формальных моделей начали появляться примерно 20 лет назад (KVEST 1997 [1], UniTESK 2002 [2], SpecExporer 2004 [3]), они не получили широкого распространения из-за значительной трудоемкости разработки формальных спецификаций, необходимых для генерации тестов.

Для разработки собственно спецификаций или моделей программ, из которых можно было бы генерировать тесты, были потрачены значительные усилия. Наиболее известными технологиями разработки спецификаций или моделей, из которых можно генерировать тесты, являются технологии SDL и UML, языки для описания тестов (или абстрактных тестов), например, TTCN-3, спецификационные расширения языков программирования JML

(Java Modeling Language [4]), Spec# [5], ACSL (ANCI/ISO C Specification Language [6]), SPARK (SPARK Ada) [7].

Сейчас ситуация в области разработки ответственного программного обеспечения меняется, вводятся новые стандарты и регламенты, которые используются при сертификации программных систем, где требования надежности, защищенности и информационной безопасности имеют первоочередное значение. Как следствие, у разработчиков доверенного ПО появляются сначала достаточно подробные спецификации интерфейсов программных систем (как на системном уровне, так и на уровне подсистем и модулей), а потом и формальные спецификации. Это позволяет по-новому взглянуть на задачи тестирования на основе формальных спецификаций или формальных моделей, так как основной барьер, мешающий распространению тестирования на основе формальных спецификация и/или моделей, постепенно преодолевается.

Уже первые проекты по внедрению технологий поддержки жизненного цикла доверенного ПО на примере верификации средств защиты операционной системы Astra Linux Special Edition [8] показали, что модели системы в реальной практике более сложны и разнообразны по сравнению с теми, которые рассматривались в первые годы появления технологий тестирования на основе моделей. Первой задачей, которая встала перед разработчиками методов и инструментов генерации тестов из формальных спецификаций в последние годы, стала задача установления отношения между требованиями к системе в целом (внешним интерфейсам) и требованиями к отдельным модулям (внутренним интерфейсам). Вторая задача состояла в том, что набор понятий, при помощи которого описываются некоторые важные свойства системы (например, свойства информационной безопасности), не совпадает и непросто отображается на понятия, в которых описываются интерфейсы реализации (в конкретном случае внешние и внутренние интерфейсы операционной системы).

Обе задачи возникают из-за того, что в большом программном проекте мы вынуждены работать с так называемыми «многоуровневыми» спецификациями. Далее в статье будут рассмотрены причины появления таких спецификаций, виды «многоуровневости» и выводы, которые приходится делать при адаптации известных схем генерации тестов и мониторинга в данном контексте.

Основными источниками введения многоуровневости спецификаций в программной инженерии являются следующие:

1) **Методология программирования «сверху-вниз»**, пошаговая детализация. Отправная точка этих методологий – инженерный опыт в технических отраслях, который сводится к выделению нескольких стадий конструирования/производства нового изделия: замысел-эскиз-макет/прототип-первая рабочая версия и так далее. Практическое преимущество такого многостадийного процесса состоит в том, что он позволяет разумно тратить усилия конструкторов, накапливать опыт и находить проблемы на ранних стадиях, экспериментировать тогда, когда эксперименты еще не слишком затратны как по средствам, так и по времени.
2) **Методы аналитической верификации**. Эти методы на практике столкнулись с чрезвычайно высокой «стоимостью» ошибок в спецификациях, в частности, из-за неправильного понимания требований к функциональности нового разрабатываемого продукта: при обнаружении таких дефектов переделка спецификации и последующая «переверификация» требовали огромных (к сожалению, часто многократно возрастающих) затрачиваемых усилий. Кроме того, ограничения инструментов верификации подталкивали строить многоуровневые модели (спецификации). В этом случае при появлении нового «уточняющего» уровня задача верификации оказывается существенно менее сложной, что упрощает работу верификатора и позволяет уложиться в ограничения инструмента верификации.
3) **Задачи анализа защищенности программных систем**. Постановка задач информационной безопасности требует тесного взаимодействия специалистов по

информационной безопасности и программистов. Привычные спецификации интерфейсов, предназначенные для описания функциональности программной системы, нацелены на описание возможностей системы, которые она должна предоставлять пользователю (например, разработчику приложений, использующих специфицируемые интерфейсы). Для специалиста по информационной безопасности главный аспект требований – это правила ограничения предоставления информации. То есть, с одной стороны, это некоторая новая группа требований, а с другой стороны, это дополнительные требования к тем же сущностям, которые были описаны в спецификациях программных интерфейсов. То есть здесь возникает необходимость «приподнять» уровень функциональных требований, чтобы в компактном и ясном виде сформулировать именно требования информационной безопасности.

4) **Задачи распространения требований системного уровня на отдельные внутренние компоненты.** Эта задача встречалась и раньше, в частности, при проектировании и верификации телекоммуникационных протоколов, но в последнее время востребованность в грамотном решении этой задачи выросла в связи с проблемами верификации (и сертификации) систем критических по надежности и безопасности. В силу того, что сами по себе системы, нуждающиеся в тщательном анализе и верификации, могут быть настолько крупными, что имеющиеся методы верификации их целиком проанализировать не позволяют, современные требования регуляторов заключаются в том, что для целевой системы сначала должен быть проведен архитектурный анализ, выделены критические компоненты (см., например, ГОСТ Р 56939-2016 «Защита информации. Разработка безопасного программного обеспечения. Общие требования»). Затем критические компоненты подвергаются настолько, насколько это возможно, тщательному анализу/верификации.

   Типичными примерами таких компонентов являются криптографические модули или модули безопасности в операционных системах и СУБД (например, Linux Security Module – LSM, в операционной системе Linux). Проблема спецификации LSM – это типичная проблема построения спецификации интерфейсов внутреннего компонента системы в соответствии с требованиями системного уровня. На системном уровне при этом следует рассматривать как минимум два уровня спецификаций. Самый верхний – это так называемая модель политики безопасности или модель управления доступом (Security Policy Model - SPM), которая сама так же может быть многоуровневой. Нижний уровень – функциональная спецификация внешних интерфейсов системы (Functional SPecification – FSP). Например, в случае ядра операционной системы, FSP – это спецификация системных вызовов.

Тем самым можно заключить, что потребность в разработке многоуровневых спецификаций возникла достаточно давно, но в последние годы в секторах, связанных с задачами обеспечения высокой надежности и защищенности программных систем, потребность уже переросла в критически важную необходимость. Требования разработки таких моделей и многоуровневых спецификаций предписываются рядом стандартов (например, ГОСТ Р ИСО/МЭК 15408-2012 и КТ-178С), где, в частности, речь идет о формальных моделях политик безопасности или о высокоуровневых и низкоуровневых спецификациях интерфейсов.

Традиционно считается, что формальная спецификация поведения системы нужна, в основном, как стартовая точка для формальной верификации системы. В действительности полезность разработки формальной спецификации не ограничивается только поддержкой формальной верификации. Разработка формальных спецификаций сама по себе положительно влияет на качество программного обеспечения, что многократно отмечалось специалистами, использовавшими формальные методы на практике – это объясняется тем, что при разработке формальной спецификации функционирование системы рассматривается в новом ракурсе. Спецификатор думает не столько о том, как можно реализовать ту или иную

функцию, сколько о том, каковы могут быть области допустимых значений входных параметров и критерии корректности реализации.

Тем не менее, весь потенциал формальных методов раскрывается тогда, когда на основе формальной спецификации проводятся различные виды верификации, включая проверку моделей (model checking) и тестирование. Необходимо заметить, что объектом верификации служит как сама модель, так и реализация, которая должна отвечать требованиям, представленными моделью/спецификацией.

Для крупных программных комплексов одним видом верификации, например, аналитической верификацией или проверкой моделей, ограничиваться нельзя в силу того, что современные методы и инструменты не справляются с такой задачей. Поэтому имеет смысл использовать спецификации для генерации тестов – такой подход называется тестированием на основе формальных моделей или на основе формальных спецификаций (Model/Specification Based Testing – MBT и SBT).

Цель данной статьи – проанализировать, какие формы многоуровневых спецификаций встречаются в реальной практике, как они используются и могут использоваться в автоматизации тестирования.

2. Виды многоуровневых моделей

Для более подробного рассмотрения методов построения многоуровневых спецификаций, рассмотрим отдельно спецификации на языках моделирования и на спецификационных расширениях языков программирования.

К первой группе относятся такие языки как ASML [9], B [10], Event-B [11], TLA+ [12], VDM [13], Z [14].

Исторически первым из перечисленных методов появился **VDM (Vienna Development Method** – Венский метод разработки программ). В настоящее время существует три диалекта нотации языка: VDM-SL, позволяющий описывать абстрактные типы данных с пред- и постусловиями функций; VDM++ расширяет VDM-SL возможностями задания объектно-ориентированных и параллельных моделей; VICE, который добавляет к VDM++ средства для описания приложений реального времени и распределенных систем.

В VDM определен ставший классическим подход к уточнению моделей. Это схема 1-to-1, то есть одной структуре данных и одной абстрактной операции должна соответствовать одна детализированная (уточненная) структура данных (тип данных) или одна операция в случае уточнения операций. Однако при этом ограничения на способ описаний уточнения очень незначительные: связь между уточненной сущностью и уточняемой задается произвольной функцией абстрагирования `retr` (от англ. retrieve – вернуться в данном случае к исходному, менее детальному описанию). Функция `retr` определяется для всех уточняемых (impl) типов данных:

```
retr: impl_Type -> abstr_Type
retr_Type_result (impl_Foo (retr__agrument)) =
  abstr_Foo (abstr_agrument)
```

Замечание. Хотя формально ограничение 1-to-1 для соответствия типов данных в VDM имеется, в случае описания соответствия сигнатур функций можно рассматривать кортеж аргументов и кортеж результатов как два (неименованных) типа данных. Это позволяет описывать отношения уточнения более гибко, чем в случае строгой трактовки 1-to-1. Но, в свою очередь, это затрудняет аналитическое доказательство соответствия. Если функция не является «чистой», то есть имеется зависимость от глобальных переменных или функция может иметь побочный эффект, это еще больше затрудняет доказательство соответствия.

На практике VDM применялся для моделирования и верификации, прежде всего, систем реального времени и встроенных систем управления. В последние годы разработчики VDM

сместили основное внимание в область моделирования киберфизических систем. Здесь также стоят задачи разработки и верификации многоуровневых моделей, но эта тематика требует отдельного исследования и не рассматривается в данной статье. Имеется список промышленных и исследовательских проектов, использовавших VDM [15].

Формальный метод B основан на теории множеств и логике предикатов первого порядка. Он поддержан коммерческим инструментом, имеется ряд примеров промышленного применения метода, в частности он использовался для разработки многоуровневой формальной модели операционной системы fmC/OS [16]. Процесс разработки был разбит на две части: предварительно была разработана абстрактная модель верхнего уровня, включающая в себя модели 8 подсистем, а затем и модель нижнего уровня, представляющая собой уточнение абстрактной модели. Далее на основе детальной модели был автоматически сгенерирован исполняемый код операционной системы. В работе утверждается, что каждый уровень модели (абстрактный и детальный) был полезен и позволил обнаружить и исправить ряд ошибок.

Формальный метод B был также использован для моделирования механизма контроля доступом в ядре ОС Linux, который включает в себя описание аппаратной изоляции (ARM TrustZone) [17]. Разработанная формальная модель является многоуровневой и состоит из четырех абстрактных машин, которые описывают процессы операционной системы, ресурсы и политику доступа. Четвертая машина включает в себя три предыдущие для получения доступа к определенным переменным и операциям. Она была частично верифицирована с использованием инструментов автоматического доказательства платформы Atelier B и инструмента проверки моделей ProB.

Примеры показывают, что метод применим и для задания абстрактной модели верхнего уровня и для детальной спецификации, а также для доказательства отношения уточнения между ними. Однако метод предлагает не очень наглядные средства для композиции моделей при помощи конструкций INCLUDES, USES, SEES, в то время как каждый модуль модели описывается в отдельном файле. Отношение уточнения может быть типа 1-to-n, [18].

Метод предполагает генерацию последовательного исходного кода на основе модели в программы на языках C, C++, Java и Ada. Наличие такой возможности является важным достоинством метода, однако в случае системного ПО, этот механизм требует специальных доработок, что при использовании импортного коммерческого продукта весьма проблематично.

Для моделирования реактивных систем широко используется модифицированная нотация, которая получила название Event-B. Она поддерживается открытыми инструментами, также позволяет проводить дедуктивную верификацию, проверку моделей, позволяет строить и верифицировать многоуровневые модели и генерировать программы на языках программирования из моделей, которые детализированы в достаточной степени.

Метод ASM (Abstract State Machines) основан на расширенных конечных автоматах и построении их композиции. Данный метод успешно применялся для моделирования и верификации программных систем, предоставляющих программный интерфейс. Есть работы по применению ASM для генерации тестов компиляторов, распределенных систем и других видов ПО.

Метод ASM предоставляет возможность разработки многоуровневых моделей и пошагового уточнения таких моделей. В работах Э. Бёргера (E. Börger) и Р. Штерка (R. Stärk) рассматривается много видов уточнения, в частности, достаточно мощный механизм для выполнения доказательства отношений уточнения, в том числе по схеме n-to-m. При уточнении в ASM удается показать не только изменение структуры данных в моделях (data refinement), но и изменение в поведении (procedural refinement).

Метод успешно позволяет моделировать параллельные системы. В то же время при помощи расширенных автоматов представляется трудным формулировать глобальные требования и

требования живости (liveness). По этой причине метод не удобен для задания абстрактных моделей верхнего уровня, но предоставляет много средств для определения детальной спецификации.

В целом можно отметить, что Microsoft Research потратило много усилий для разработки инструмента MBT на основе ASM (так называемый ASMT). Вместе с тем, через 2 года экспериментов от нотации ASML пришлось отказаться, так как она оказалась неудобной для программистов-практиков. Кроме того, из публикаций видно, что для тестирования на основе моделей многоуровневые спецификации не применялись.

Инструменты на основе ACSL-спецификаций. В первую очередь это инструменты Frama-C для дедуктивной верификации C-программ. Имеется также расширение E-ACSL – Executable ACSL – Исполнимое подмножество ACSL, которое используется для спецификации проверочных утверждений (assertions) при тестировании и мониторинге C-программ. ACSL не предлагает никакой общей схемы уточнения и построения многоуровневых спецификаций, но в рамках данной работы он интересен тем, что предлагает некоторый единый язык спецификаций, который можно использовать и для аналитической верификации, и для тестирования.

Вывод. На основе рассмотренных работ можно заключить, что специалисты по формальным методам активно используют многоуровневые модели в аналитической верификации, но не пользуются такими моделями при тестировании на основе моделей. Это, по-видимому, объясняется чрезмерным усложнением системы тестирования и отсутствием принципиально нового качества, которого авторы публикаций пока не видят при тестировании на основе многоуровневых моделей.

3. Использование многоуровневых моделей для тестирования и мониторинга

Вместе с тем опыт построения тестовой системы для проверки средств защиты информации (СЗИ) операционной системы Linux [8] показал, что потенциал многоуровневых моделей может быть использован в более полной мере. Этот подход, в частности, может быть использован при организации тестирования системного интерфейса операционной системы (системные вызовы – system calls) на соответствие модели управления доступом и при тестировании внутренних модулей системы на проверку соответствия требованиям, сформулированным на более высоких уровнях, например, на уровне той же модели управления доступом или на уровне системных вызовов. Заметим, что роль высокоуровневых моделей при организации тестирования может быть различной. Такие модели могут использоваться для:

- фильтрации (отбраковки) некорректных входных тестовых данных – на основе предусловий;
- генерации тестового оракула (автоматического анализатора полученного результата) - на основе постусловия;
- оценки полученного тестового покрытия (на основе структуры пред- и постусловий);
- построения модели тестовых последовательностей, например, так, как это делается в UniTESK.

Данная работа не ставит своей целью дать детальное описание тестовой системы, работающей с многоуровневыми спецификациями. Здесь мы опишем только общую схему построения такой системы.

На рис. 1 показана достаточно простая схема использования двухуровневой спецификации. Модель в данном случае используется для определения (и автоматической генерации) «тестового оракула» – программного компонента системы тестирования, который в динамике анализирует корректность поведения тестируемой системы. В данном случае оракул проверяет, соответствует ли результат очередного системного вызова спецификации этого

вызова (Оракул Д). В дополнение к Оракулу Д на основе Модели управления доступом можно построить Оракул А. Вместе с тем, даже в такой простой ситуации построение Оракула А может оказаться сложной задачей, а поскольку оракул работает во время тестирования, его вычислительная сложность и потребности в ресурсах памяти может оказаться чрезмерными. По этой причине в промышленном проекте по верификации средств защиты информации операционной системы Astra Linux Special Edition от Оракула А пришлось отказаться – на рисунке этот оракул изображен при помощи пунктирной линии.

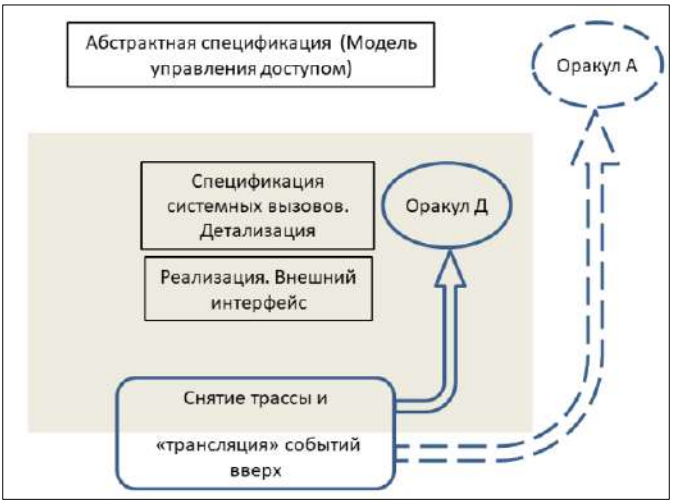

Рис. 1. Схема мониторинга и анализа трассы на уровне системных вызовов ОС
Fig. 1. Trace monitoring and analysis scheme at the level of OS system calls

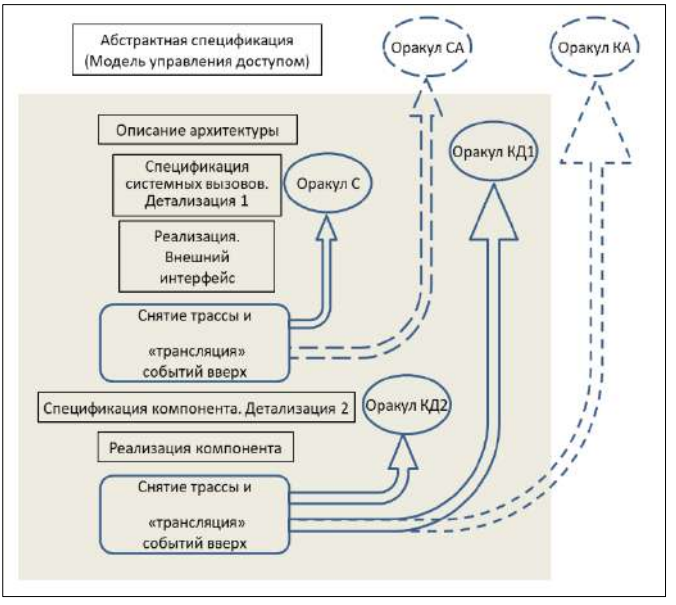

Рис. 2. Схема мониторинга и анализа трассы на уровне системных вызовов и внутреннего компонента ОС
Fig. 2. Trace monitoring and analysis scheme at the level of system calls and internal OS component

Более сложная схема использования многоуровневых моделей показана на рис. 2. Здесь целевыми объектами тестирования является ядро ОС (системные вызовы) и внутренний компонент ядра. В случае защищенной операционной системы семейства Linux, это может быть так называемый модуль безопасности (Linux Security Module – LSM). Функциональность LSM определяется Моделью управления доступом, но является абстрактной по отношению к спецификации интерфейса LSM (то есть не содержит всех деталей, необходимых для спецификации функций LSM. В частности, в модели не описаны сложные структуры данных, с которыми работают программы ядра ОС, а они являются важной составляющей описания поведения LSM.

На рис. 2 также два оракула отмечены пунктирными линиями – эти оракулы реализовать крайне сложно.

Важно отметить, что причиной повышенной сложности построения оракулов в приведенных примерах является сложность отношения уточнения, которая выбирается в том или ином случае, а сложность отношения в свою очередь зависит от степени различия интерфейсов моделей и реализаций. Можно сказать, что всегда, когда выбран (или пришлось выбрать) механизм уточнения типа 1-to-n или n-to-m, оракул оказывается слишком сложным. Опыт ИСП РАН по верификации моделей управления доступом и ядер операционных систем показал, что имеется два способа отказаться от прямолинейного решения динамической проверки на основе многоуровневых спецификаций. Первый способ был использован в проекте по верификации ОС Astra Linux Special Edition [19]. При разработке спецификаций системных вызовов статически было доказано, что выполнение требований функциональной спецификации системных вызовов влечет выполнение требований безопасности, сформулированных в Модели управления доступом. Это означает, что при динамической проверке выполнения системных вызовов (при анализе трассы, где сохраняется информация об аргументах и результатах системных вызовов) положительный вердикт Оракула Д (рис. 1) гарантирует выполнения требований Модели управления доступом. Эта схема была выбрана по той причине, что Модель и спецификация были связаны отношением n-to-m.

При разработке Модели управления доступом для ОС Linux Альт 8 СП было принято решение приблизить структуру интерфейсов Модели к структуре системных вызовов. Это существенно упростило отношение уточнения, по этой причине тестирование в данном случае можно организовать с использованием Оракула А, что не будет являться слишком сложной задачей [20].

При тестировании корректности использования внутренних компонентов системы на соответствие общесистемным требованиям необходимо помимо спецификации внешнего (наблюдаемого извне) поведения системы иметь в распоряжении спецификацию архитектуры реализации и значимых протоколов взаимодействия внутренних компонентов. Так в случае модуля LSM важным требованием безопасности является вызов соответствующей функции LSM всегда, когда операционная система разрешает доступ к некоторому объекту (запретить доступ операционная система иногда может и без обращения к LSM). Такого рода протоколы либо должны включаться в спецификацию архитектуры, либо извлекаться из спецификации архитектуры, желательно, простым способом. На примере, приведенном на рис. 2, показано, что Оракул КД2 определяется на основе спецификации архитектуры и спецификации системных вызовов.

4. Заключение

В статье представлены результаты первого года работ по гранту РФФИ 20-01-00568 по теме «Мониторинг и тестирование модулей операционных систем на основе абстрактных моделей поведения системы». Были исследованы различные способы построения и верификации многоуровневых формальных моделей (спецификаций) программ и проанализированы проблемы использования таких спецификаций для целей тестирования и мониторинга. На

основе обобщения опыта работ ИСП РАН по верификации средств защиты информации операционных систем и других программных комплексов были сформулированы основные принципы построения тестовой системы на основе многоуровневых спецификаций.

Для апробации предложенного подхода мы разработали экспериментальную систему тестирования на основе многоуровневых спецификаций. В настоящее время удалось решить следующие задачи.

1. разработан метод уточнения на основе состояний для формального метода Event-B. Данный метод позволяет упростить процессы разработки и верификации многоуровневых спецификаций в ситуациях, когда интерфейсы различных уровней спецификации существенно отличаются друг от друга;
2. с использованием предложенного метода разработана многоуровневая спецификация, которая может быть использована для демонстрации возможностей экспериментальной системы тестирования;
3. разработан компонент мониторинга системных вызовов и внутренних событий ядра Linux на основе kprobes/kretprobes. Система включает в себя модуль ядра для мониторинга и приложение-демон пользовательского пространства для сбора информации от модуля ядра;
4. разработаны компоненты для сбора начального состояния операционной системы перед выполнением тестов, для преобразования трасс событий, снятых с ядра Linux, в вид пригодный для воспроизведения на многоуровневой спецификации из пункта 2;
5. на основе ProB разработан инструмент воспроизведения трасс из пункта 4 на многоуровневой спецификации из пункта 2;
6. на основе знаний о системе Linux, многоуровневой формальной спецификации из пункта 2, включающей в себя уровни дискретного контроля доступа (DAC), списков управления доступом (ACL), подсистемы контроля целостности (IMA/EVM), мандатного контроля доступа, разработана методика построения тестового набора для их проверки на реальной системе;
7. в соответствии с методикой построения тестового набора создан тестовый пакет программ, покрывающих разные виды доступа, а также перечисленные в пункте 6 системы контроля доступа;
8. разработанные компоненты из пунктов 2, 3, 4, 5 и 7 объединены в единую схему запуска тестов на системе GNU/Linux, снятия трасс событий с этих тестов, трансляции трасс в вид, пригодный для воспроизведения на многоуровневой спецификации, и, собственно, воспроизведения оттранслированных трасс на многоуровневой формальной спецификации для проверки соответствия реального поведения системы с модельным в рамках экспериментального программного комплекса.

Развитие предложенного подхода планируется направить на пополнение техник мониторинга техниками генерации тестов на основе многоуровневых спецификаций.

Список литературы / References

[1] I. Burdonov, A. Kossatchev, A.K. Petrenko, D. Galter. Kvest: Automated generation of test suites from formal specifications. Lecture Notes in Computer Science, vol. 1708, 1999, pp. 608-621.

[2] I.B. Bourdonov, A.S. Kossatchev, V.V. Kuliamin, A.K. Petrenko. UniTesK test suite architecture, Lecture Notes in Computer Science, vol. 2391, 2002, pp. 77-88.

[3] SpecExporer. URL: https://docs.microsoft.com/ru-ru/archive/msdn-magazine/2013/december/model-based-testing-an-introduction-to-model-based-testing-and-spec-explorer, accessed 21.12.2020.

[4] JML (Java Modeling Language), URL: http://www.eecs.ucf.edu/~leavens/JML//index.shtml, accessed 21.12.2020.

[5] M. Barnett, K. Rustan M. Leino, and Wolfram Schulte. The Spec# programming system: An overview. Lecture Notes in Computer Science, vol. 3362, 2004, pp. 49-69.

[6] ACSL (ANCI/ISO C Specification Language). https://www.academia.edu/16532644/ACSL_ANSI_ISO_C_Specification_Language, accessed 21.12.2020.
[7] SPARK. http://www.spark-2014.org, accessed 21.12.2020.
[8] П.Н. Девянин, Д.В. Ефремов, В.В. Кулямин, А.К. Петренко, А.В. Хорошилов, И.В. Щепетков. Моделирование и верификация политик безопасности управления доступом в операционных системах. М., Горячая линия – Телеком, 2019 г., 214 стр. / P.N. Devyanin, D.V. Efremov, V.V. Kulyamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Modeling and verification of access control security policies in operating systems. M., Hotline – Telecom, 2019, 214 p. (in Russian).
[9] Y. Gurevich. Evolving algebras 1993: Lipari guide. In: Börger E. (ed.) Specification and Validation Methods, pp. 9-36. Oxford University Press, Oxford, 1995.
[10] J.-R. Abrial. The B-Book: Assigning Programs to Meanings. Cambridge University Press, 2005, 815 p.
[11] J.-R. Abrial. Modeling in Event-B: System and Software Engineering. Cambridge University Press, 2010, 612 p.
[12] L. Lamport. Specifying Concurrent Systems with TLA+. In: Broy M., Steinbrüggen R. (eds). Calculational System Design, pp. 183–247. IOS Press, Amsterdam, 2000.
[13] D. Bjorner, C.B. Jones. The Vienna Development Method: The Meta-Language. Lecture Notes in Computer Science, vol. 61, 1978.
[14] J.-R. Abrial. Data Semantics. In Proc. of the IFIP Working Conference on Data Base Management, 1974, pp. 1–59.
[15] C.B. Nielsen, P.G. Larsen, J. Fitzgerald, J. Woodcock. Systems of systems engineering: basic concepts, model-based techniques, and research directions. ACM Computing Surveys (CSUR), vol. 48, issue 4, 2015, article no. 18.
[16] C. Danmin, S. Yue, C. Zhiguo. A Formal Specification in B of an Operating System. The Open Cybernetics & Systemics Journal, no. 9, 2015, pp. 1124-1129.
[17] L. Ren, R. Chang, Q. Yin, W. Wang. Using the B Method to Formalize Access Control Mechanism with TrustZone Hardware Isolation. Lecture Notes in Computer Science, vol. 10701, 2017, pp. 759-769.
[18] S. Hallerstede. M. Hasangic, S. Krings, P.G. Larsen, M. Leuschel. From Software Specifications to Constraint Programming. Lecture Notes in Computer Science, vol. 10886, 2018, pp. 21-36.
[19] D. Efremov, I. Shchepetkov. Runtime Verification of Linux Kernel Security Module. Lecture Notes in Computer Science, vol. 12233, 2020, pp. 185-199.
[20] V. Kuliamin, A. Khoroshilov, D. Medvedev. Formal Modeling of Multi-Level Security and Integrity Control Implemented with SELinux. In Proc. of the 2019 Actual Problems of Systems and Software Engineering (APSSE), 2019, pp. 131-136.

Информация об авторах / Information about authors

Александр Константинович ПЕТРЕНКО – доктор физико-математических наук, профессор, начальник отдела ИСП РАН, профессор МГУ и ВШЭ. Область интересов: формальные методы программной инженерии, тестирование программного и аппаратного обеспечения, формальная спецификация требований.

Alexander Konstantinovich PETRENKO – Doctor of Physical and Mathematical Sciences, Professor, Head of the Department of ISP RAS, Professor of Moscow State University and Higher School of Economics. Areas of interest: formal methods of software engineering, testing of software and hardware, formal specification of requirements.

Денис Валентинович ЕФРЕМОВ – младший научный сотрудник ИСП РАН. Основные научные интересы: формальная верификация, статический и динамический анализ операционных систем, инфраструктура разработки операционных систем.

Denis Valentinovich EFREMOV – Junior Researcher at ISP RAS. Areas of Interest: formal verification, static and dynamic analysis of operating systems, operating systems development infrastructure.

Евгений Валерьевич КОРНЫХИН – кандидат физико-математических наук, доцент кафедры системного программирования МГУ, старший научный сотрудник ИСП РАН. Область интересов: формальная дедуктивная верификация моделей, тестирование на основе моделей.

Eugeny Valerievich KORNYKHIN – Ph.D. in Physics and Mathematics, Associate Professor of system programming departments at Moscow State University and Senior Researcher at ISP RAS. Fields of Interest: formal deductive verification, model-based testing.

Виктор Вячеславович КУЛЯМИН – кандидат физико-математических наук, ведущий научный сотрудник ИСП РАН, доцент кафедр системного программирования МГУ и ВШЭ. Область интересов: формальная дедуктивная верификация моделей, тестирование на основе моделей.

Viktor Vyacheslavovich KULIAMIN – Ph.D. in Physics and Mathematics, leading researcher at ISP RAS, associate professor of system programming departments at Moscow State University and the Higher School of Economics. Fields of Interest: formal deductive model verification, model-based testing

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

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

Илья Викторович ЩЕПЕТКОВ – стажер-исследователь. Область интересов: разработка и верификация формальных моделей компьютерных систем

Ilya Viktorovich SHCHEPETKOV – Intern Researcher. Fields of Interest: development and verification of formal models of computer systems

DOI: 10.15514/ISPRAS–2020–32(6)–2

Верификация соответствия между разноуровневыми моделями функциональных требований

А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1
НИУ Высшая школа экономики,
101978, Россия, г. Москва, ул. Мясницкая, д. 20
Московский физико-технический институт,
141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. В статье предлагаются методы доказательства соответствия между разноуровневыми моделями, нацеленные на доказательство свойств безопасности в индивидуальной трассовой семантике. Эти методы более просты и пригодны для применения при решении практических задач верификации сложных видов функциональных требований по сравнению с традиционными подходами, основанными на установлении отношения уточнения между моделями.

Ключевые слова: формальная верификация; формальные модели; трассовая семантика.

Для цитирования: Хорошилов А.В. Верификация соответствия между разноуровневыми моделями функциональных требований. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 19-30. DOI: 10.15514/ISPRAS–2020–32(6)–2

Благодарности. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00954.

Verification of Compliance for Multilevel Models in Individual Trace Semantics

A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia,
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia
National Research University, Higher School of Economics
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
Moscow Institute of Physics and Technology (MIPT)
141700, Russia, Moscow region, Dolgoprudny, Campus per., 9

Abstract. The paper considers the problem of verification of compliance between models representing the same system on different level of abstraction. The existing approaches are mostly based on refinement relation. But the models representing industrial systems are quite big and complex, while semantics gap between the level is

quite big. As a result, the existing methods became too complex and labour intensive. The paper presents new verification techniques that targets to prove multimodel compliance in terms of individual trace semantics. The techniques assume that each model is verified, i.e. it is proved that starting from initial states of labelled transition system is not possible to reach unsafe states by using valid transitions. The first proposed technique allows to prove that the detailed model satisfies to requirements of the abstract model, i.e. reachable states of detailed model do not include states corresponding to unsafe states of the abstract model. The second proposed technique allows to prove that the detailed model satisfies to behaviour specification of the abstract model, i.e. all reachable transitions of the detailed model do not include transitions corresponding to invalid transitions of the abstract model. For each technique the correspondence relation is defined in terms of the models, i.e. the relations are formally defined and they can be used for analysis with interactive or automated provers. At the same time, there are some requirements to that relations that are expressed in terms of low level events that exist hypothetically only and can be analyzed theoretically only. As a result, the proposed techniques provides a reasonable approach to prove compliance between mulilevel models in more approachable way for industrial settings.

Keywords: formal verification; formal models; trace semantics

For citation: Khoroshilov A.V. Verification of compliance for multilevel models in individual trace semantics. Trudy ISP RAN/Proc. ISP RAS, vol. 1, issue 2, 2017. pp. 19-30 (in Russian). DOI: 10.15514/ISPRAS-2020-32(6)-2

Acknowledgements. The reported study was funded by RFBR, project number 20-07-00954.

1. Введение

Формальные модели программных систем являются одним из механизмов повышения доверия к безопасности и надёжности программных систем, решающих ответственные задачи, такие как защита информацией и управление доступом в информационно-вычислительных системах в условиях враждебной среды, управление ответственным оборудованием и т.д. Во многих случаях для одной программной системы разрабатывается несколько формальных моделей, описывающих разные аспекты её функционирования на разных уровнях детализации. Эти модели при описании практически значимых программных систем являются достаточно большими и сложными, а семантический разрыв между моделями разных уровней оказывается достаточно большой, чтобы была возможность эффективно применять существующие методы верификации соответствия между разноуровневыми моделями. Основная причина этого заключается в объективной сложности данной задачи в её полной постановке.

В рамках настоящей работы предлагаются методы доказательства соответствия между разноуровневыми моделями, нацеленные на доказательство свойств безопасности в индивидуальной трассовой семантике. Эти методы более просты и пригодны для применения при решении практических задач верификации сложных видов функциональных требований по сравнению с традиционными подходами, основанными на установлении отношения уточнения между моделями.

Дальнейшее изложение материала организовано следующим образом. В разд. 2 вводятся базовые термины, используемые далее. Разд. 3 содержит описание предлагаемых методов и доказательства их корректности. В разд. 4 обсуждаются вопросы практического применения методов и направления дальнейших исследований. В заключении приводятся краткие выводы по результатам работы.

2. Предварительные определения

В рамках данной работы целевой системой мы будем называть программную или программно-аппаратную систему, которая подлежит моделированию и верификации.

2.1 Формализация поведения целевой системы

Прежде чем рассматривать модели целевой системы, с которыми выполняется непосредственная работа по анализу и верификации, введём понятие низкоуровневой модели поведения целевой системы, которая предоставит аппарат для дальнейших рассуждений о функциональных свойствах её поведения.

Определение 1. Обозначим **LLEvents** множество всех потенциально возможных низкоуровневых событий, из который состоит функционирование целевой системы. Состав событий может зависеть от природы целевой системы. Например, для операционных систем такими событиями могут быть события аппаратного уровня, такие как изменение значения регистров, поступление прерываний и т.д. Предполагается, что элементы этого множества представляют собой не только тип события, но все его сопутствующие атрибуты, включая его пространственные и временные характеристики, поэтому без ограничения общности будем считать, что любые два возможных события в аппаратуре представляются различными элементами **LLEvents**. Также будем считать, что на множестве **LLEvents** задано антирефлексивное бинарное отношение $\prec_{dep}$, которое означает, что второе событие причинно зависит от первого события явным или неявным образом, в частности, второе событие может произойти по времени только строго позже первого.

Определение 2. Трассой низкоуровневых событий будем называть конечное или бесконечное финитарное частично-упорядоченное множество событий $(\mathbf{T}, \preccurlyeq)$, где

- $\mathbf{T} \subseteq \mathbf{LLEvents}$;
- частичный порядок $\preccurlyeq$ является транзитивным и рефлексивным замыканием $\prec_{dep}$, т.е. $\preccurlyeq = \left(\{(e,e)|e \in \mathbf{T}\} \cup \prec_{dep|T}\right)^*$ и определяет порядок событий во времени и их причинную зависимость;
- $\prec_{dep|T} = \prec_{dep} \cap\ \mathbf{T} \times \mathbf{T}$ — ограничение отношения $\prec_{dep}$ на множество $\mathbf{T}$;
- свойство финитарности означает, что $\forall\, e \in \mathbf{T}$ множество $\{e' \in \mathbf{T} \mid e' \preccurlyeq e\}$ является конечным.

Далее будем использовать следующие обозначения:

- $\wp(\mathbf{X})$ – множество всех подмножеств множества $\mathbf{X}$;
- $\wp_{\mathrm{fpo}}(\mathbf{X})$ – множество всех финитарных частично-упорядоченных подмножеств множества $\mathbf{X}$;
- $\mathbf{Traces}\left(\mathbf{LLEvents}, \prec_{dep}\right) \subseteq \wp_{\mathrm{fpo}}(\mathbf{LLEvents})$ – множество всех трасс событий над множеством **LLEvents** с отношением зависимости $\prec_{dep}$;
- $\wp_{\mathrm{directed}}(\mathbf{X}, \preccurlyeq) = \{\mathbf{X}' \subseteq \mathbf{X} \mid \forall\, \mathbf{T}_1, \mathbf{T}_2 \in \mathbf{X}'\ \exists\, \mathbf{T}_3 \in \mathbf{X}' \colon \mathbf{T}_1 \preccurlyeq \mathbf{T}_3 \wedge \mathbf{T}_2 \preccurlyeq \mathbf{T}_3)\}$ – множество всех направленных подмножеств частично-упорядоченного множества $\mathbf{X}$;
- $\wp_{\mathrm{dcpo}}(\mathbf{X}, \preccurlyeq) = \{\, \mathbf{X}' \in \wp(\mathbf{X}) \mid \forall\, \mathbf{Y} \in \wp_{\mathrm{directed}}(\mathbf{X}', \preccurlyeq|_{\mathbf{X}'})\ \mathbf{sup}(\mathbf{Y}) \in \mathbf{X}'\}$ – множество всех dcpo-подмножеств множества $\mathbf{X}$, т.е. подмножеств, являющихся направленными полными частично-упорядоченными множествами;
- $\mathbf{Prefix}(\mathbf{T}, \preccurlyeq) = \{(\mathbf{T}', \preccurlyeq') | \mathbf{T}' \subseteq \mathbf{T} \wedge \forall e \in \mathbf{T}\, \forall\, e' \in \mathbf{T}'\, (e \preccurlyeq e') \Rightarrow e \in \mathbf{T}' \wedge \preccurlyeq' = \preccurlyeq|_{\mathbf{T}'}\}$ – множество всех префиксов частично-упорядоченного множества $(\mathbf{T}, \preccurlyeq)$, а $\mathbf{Prefix}^{\mathrm{fin}}(\mathbf{T}, \preccurlyeq)$ – множество всех конечных префиксов;
- $\preccurlyeq_{\mathrm{pre}} = \{(\mathbf{T}_1, \mathbf{T}_2) \mid \mathbf{T}_1 \in \mathbf{Prefix}(\mathbf{T}_2)\}$ – отношение «является префиксом» на множестве частично-упорядоченных множеств;
- $\wp_{\mathrm{cleft}}(\mathbf{X}) = \left\{\mathbf{X}' \in \wp_{\mathrm{dcpo}}\left(\mathbf{X}, \preccurlyeq_{\mathrm{pre}}\right) \middle| \forall (\mathbf{T}, \preccurlyeq) \in \mathbf{X}'\ \mathbf{Prefix}(\mathbf{T}, \preccurlyeq) \subseteq \mathbf{X}'\right\}$ — множество всех **dcpo**-подмножеств $\mathbf{X}$ относительно порядка $\preccurlyeq_{\mathrm{pre}}$, являющихся замкнутыми влево относительно частичных порядков своих элементов, где $\mathbf{X}$ – некоторое множество частично-упорядоченных множеств.

Определение 3. Обозначим $\mathbf{LLTraces} \in \wp_{\text{cleft}}(\mathbf{Traces}(\mathbf{LLEvents}, \prec_{dep}))$ множество потенциально возможных трасс низкоуровневых событий. Требование потенциальной возможности трассы подразумевает, что трасса является согласованной с правилами корректного поведения окружения целевой системы, то есть, например, в ней отсутствуют подмножества событий, возможные только в случае сбоев в аппаратуре.

В рамках данных определений любая целевая система характеризуется элементом множества $\wp_{\text{cleft}}(\mathbf{LLTraces})$, обозначаемым **LLModel**, который представляет собой множество трасс, в которых целевая система может участвовать при всех возможных сценариях развития событий.

Любое требование к целевой системе может быть представлено в виде подмножества $\wp_{\text{cleft}}(\mathbf{LLTraces})$, то есть множества множеств трасс, которые являются удовлетворяющими данному требованию.

В рамках данной работы мы ограничимся рассмотрением требований безопасности в индивидуальной трассовой семантике [1] (т. е. требований подкласса $\mathbf{S_1L_0}$ класса $\mathbf{HN}_1$ [2]), которые задаются множеством корректных трасс $\mathbf{LLSafe} \subseteq \mathbf{LLTraces}$, таким что $\forall\, t \in \mathbf{LLTraces}\ t \notin LLSafe \Rightarrow \exists t_m \in \mathbf{Prefix}^{\text{fin}}(t)\colon \forall\, t' \in \mathbf{LLTraces}\ t_m \in \mathbf{Prefix}^{\text{fin}}(t') \Rightarrow t' \notin \mathbf{LLSafe}$. Другими словами, целевая система удовлетворяет такому требованию, если **LLModel** является подмножеством **LLSafe**, а множество **LLSafe** характеризуется тем, что оно запрещает наличия в трассе «плохих событий», то есть появление такого события относит трассу к числу недопустимых.

Класс требований безопасности достаточно широк, например, в него входят требования, которые выражаются в виде предусловий и постусловий операций, а также требования отсутствия в трассе недопустимых событий. Среди требований, которые не входят в этот класс, можно привести, например, следующие:

- требования живучести, то есть требования, которые предписывают бескрайнего появления «хороших событий»;
- требования существования, то есть требующие наличия среди всех возможных трасс выполнения трасс с определёнными свойствами;
- требования детерминизма, например, требующие, чтобы определённая функция в определённых условиях возвращала одно из множества допустимых значений, но для фиксированной целевой системы это значение всегда было одним и тем же.

2.2 Абстрактные модели целевой системы

Поскольку низкоуровневая модель поведения целевой системы слишком детальна и сложна, то на практике для верификации используются более абстрактные модели. Одним из наиболее распространённых механизмов описания таких моделей являются автоматные модели, которые для нашего рассмотрения будут представляться как системы размеченных переходов.

Определение 4. Система размеченных переходов **LTS** — это четвёрка $(S, S_{\text{init}}, L, \rightarrow)$, где

- S – множество состояний;
- $S_{\text{init}} \subseteq S$ – непустое множество начальных состояний;
- L – множество меток;
- $\rightarrow \subseteq S \times L \times S$ – множество переходов между состояниями, помеченных метками.

Определение 5. Трассой **LTS** называется конечная или бесконечная последовательность переходов, такая что

- если последовательность не пустая, то начальное состояние первого перехода принадлежит $\underline{S_{\text{init}}}$,
- начальное состояние i-го перехода в последовательности совпадает с конечным состоянием $(i-1)$-го перехода для $i > 1$.

Множество всех возможных трасс системы размеченных переходов **LTS** будем обозначать **LTSTraces**(**LTS**). Когда из контекста понятно о какой системе размеченных переходов идёт речь, будет использоваться сокращённое обозначение **LTSTraces**. Множество состояний, входящих в переходы из трассы **T**, будем обозначать **states**(**T**), а множество конечных состояний переходов из трассы **T**, будем обозначать **endstates**(**T**).

При построении **LTS** для моделирования поведения целевой системы любой трассе низкоуровневых событий $\mathbf{T}_{\mathrm{LL}} \in \mathbf{LLTraces}$ соответствует $\alpha(\mathbf{T}_{\mathrm{LL}}) \in \mathbf{LTSTraces}$, которое представляет собой трассу, задающую модельное поведение, представленное в терминах модели **LTS**. При этом для $\alpha(\mathbf{T}_{\mathrm{LL}})$ требуется, чтобы $\forall\, \mathbf{T'}_{\mathrm{LL}} \in \mathbf{Prefix}(\mathbf{T}_{\mathrm{LL}})\ \alpha(\mathbf{T'}_{\mathrm{LL}})$ являлась префиксом последовательности $\alpha(\mathbf{T}_{\mathrm{LL}})$. Также предполагается, что конечным трассам низкоуровневых событий **TLL** соответствуют конечные трассы абстрактных событий $\alpha(\mathbf{T}_{\mathrm{LL}})$.

Иногда при определении функции α поддерживается формирование специальных значений, обозначающих невозможность построения абстракции для данной низкоуровневой трассы. В рамках данной работы для упрощения изложения считается, что при необходимости в абстрактной модели присутствуют элементы, явным образом моделирующие такие ситуации.

LTS, как правило, описывается на том или ином языке спецификаций, а отображение α остаётся в голове у автора спецификации и используется им для оценки адекватности разработанной **LTS**. Поэтому хорошей практикой является стараться, чтобы α была устроена достаточно прямолинейно, чтобы избежать ошибок в формулировке **LTS**, которые не могут быть выявлены инструментами из-за отсутствия формальной модели низкого уровня.

Разработка модели **LTS** проводится с двумя основными целями:

- получения чёткой, однозначно интерпретируемой формулировки модели поведения целевой системы на заданном уровне абстракции и выявления за счёт этого целого ряда проблем, связанных с внутренними противоречиями и неоднозначностями в понимании требований к системе [3];
- доказательство корректности модели поведения относительно требований, заданных в терминах той же модели **LTS**.

Для выражения требований к целевой системе могут быть использованы, например, формулы в той или иной темпоральной логике, использующей **LTS** в качестве основы для означивания пропозициональных переменных. Другой вариант заключается в описании требований в виде инвариантов на состоянии модели **LTS**. Далее рассмотрим подробнее второй вариант, поскольку он чаще возникает в нашей практике верификации функциональных требований к системам защиты информации.

Итак, для целей верификации дополнительно к базовой модели **LTS** добавляются два элемента $(S_{\mathrm{safe}}, \rightarrow_{\mathrm{safe}})$, где

- $S_{\mathrm{safe}} \subseteq S$ – множество безопасных состояний;
- $\rightarrow_{\mathrm{safe}} \subseteq \rightarrow$ – множество корректных переходов.

В качестве цели для доказательства формулируется утверждение, что множество достижимых состояний из начальных состояний S_{init} по переходам из множества $\rightarrow_{\mathrm{safe}}$ невозможно попасть в небезопасное состояние из $S_{\mathrm{unsafe}} = S \setminus S_{\mathrm{safe}}$. Доказательство, как правило, проводится по индукции через утверждения вида $S_{init} \subseteq S_{\mathrm{safe}}$ и $\rightarrow_{\mathrm{safe}} [S_{\mathrm{safe}}] \subseteq S_{\mathrm{safe}}$, то есть что любое начальное состояние является безопасным и любой корректный переход из безопасного состояния ведёт в безопасное состояние.

В терминах низкоуровневых поведений модель поведения корректно реализованной целевой системы задаётся при помощи множества корректных трасс:

$$\mathbf{LLSafeT}_{\mathrm{LTS}} = \mathbf{LLTraces} \setminus$$
$$\{\mathbf{T}_{\mathrm{LL}} \in \mathbf{LLTraces} \mid \exists\ \mathbf{T'}_{\mathrm{LL}} \in \mathbf{Prefix}^{\mathrm{fin}}(\mathbf{T}_{\mathrm{LL}}) \colon \alpha(\mathbf{T'}_{\mathrm{LL}}) \cap \rightarrow_{\mathrm{unsafe}} \neq \emptyset\},$$

где $\rightarrow_{\text{unsafe}} = \rightarrow \setminus \rightarrow_{\text{safe}}$, а запись $\alpha(\mathbf{T}'_{\text{LL}}) \cap \rightarrow_{\text{unsafe}} \neq \emptyset$ обозначает отсутствие некорректных переходов в трассе $\alpha(\mathbf{T}'_{\text{LL}})$.

Соответственно, доказываемое требование задаётся при помощи множества корректных трасс:

$\mathbf{LLSafeS}_{\text{LTS}} = \mathbf{LLTraces} \setminus$

$\{\mathbf{T}_{\text{LL}} \in \mathbf{LLTraces} \mid \exists\ \mathbf{T}'_{\text{LL}} \in \mathbf{Prefix}^{\text{fin}}(\mathbf{T}_{\text{LL}})\colon \mathbf{states}(\alpha(\mathbf{T}'_{\text{LL}})) \cap S_{\text{unsafe}} \neq \emptyset\}$.

Таким образом, результатом верификации **LTS** модели является следующее утверждение: Если целевая система удовлетворяет требованию $\mathbf{LLSafeT}_{\text{LTS}}$, то она удовлетворяет и требованию $\mathbf{LLSafeS}_{\text{LTS}}$, или $\mathbf{LLSafeT}_{\text{LTS}} \subseteq \mathbf{LLSafeS}_{\text{LTS}}$.

3. Метод доказательства соответствия между разноуровневыми моделями

Достаточно распространённой ситуацией является появление нескольких абстрактных моделей, описывающих целевую систему на разных уровнях детализации. При этом возникает естественное желание доказать, что более детальная модель удовлетворяет требованиям более абстрактной. Традиционный подход к решению данной задачи заключается в использовании отношения уточнения между моделями [4], включая различные вариации, допускающие трансформацию алфавита моделей [5], введение дополнительных действий на детальном уровне [6, 7], а также разбиение абстрактных действий на множество детальных [8, 9], в том числе в более сложных соотношениях (n к m) [10, 11].

Тем не менее, на практике встречаются случаи, где применение даже наиболее гибких методов установления отношения уточнения оказывается неприемлемо сложным и трудоёмким, поэтому возникает потребность в более эффективных методах установления соответствия между разноуровневыми моделями.

3.1 Доказательство соответствия свойствам безопасности

Рассмотрим, как эта проблема может быть рассмотрена с формальной точки зрения в терминах трассовой семантики.

Предположим, что более абстрактная модель целевой системы задана системой размеченных переходов **LTS1** $(S_1, S_{\text{init1}}, L_1, \rightarrow_1)$, множеством безопасных состояний S_{safe1} и множеством корректных переходов $\rightarrow_{\text{safe1}}$, а более детальная модель – системой размеченных переходов **LTS2** $(S_2, S_{\text{init2}}, L_2, \rightarrow_2)$, множеством безопасных состояний S_{safe2} и множеством корректных переходов $\rightarrow_{\text{safe2}}$.

Тогда первое желаемое свойство для доказательства соответствия между моделями заключается в том, чтобы целевые системы, описываемые более детальной моделью, удовлетворяли свойствам безопасности, доказанным для абстрактной модели. Формально это можно сформулировать так: Если целевая система удовлетворяет требованию $\mathbf{LLSafeT}_{\text{LTS2}}$, то она удовлетворяет и требованию $\mathbf{LLSafeS}_{\text{LTS1}}$ или $\mathbf{LLSafeT}_{\text{LTS2}} \subseteq \mathbf{LLSafeS}_{\text{LTS1}}$.

Определение 6. Для $\mathbf{T}_{\text{LL}} \in \mathbf{LLTraces}^{\text{fin}}$ обозначим $\mathbf{newtrace}(\mathbf{T}_{\text{LL}}, \alpha)$ хвост трассы $\alpha(\mathbf{T}_{\text{LL}})$, такой что он не появлялся ни в одной трассе $\alpha(\mathbf{T}'_{\text{LL}})$ ни для одного $\mathbf{T}'_{\text{LL}} \in \mathbf{Prefix}(\mathbf{T}_{\text{LL}})$. Соответственно, обозначим $\mathbf{newstates}(\mathbf{T}_{\text{LL}}, \alpha)$ множество $\mathbf{endstates}(\mathbf{newtrace}(\mathbf{T}_{\text{LL}}, \alpha))$, если $\mathbf{newtrace}(\mathbf{T}_{\text{LL}}, \alpha)$ не пусто, и пустое множество в противном случае.

Определение 7. Отношением соответствия между состояниями моделей **LTS1** и **LTS2** будем называть отношение $\mathbf{R} \subseteq S_1 \times S_2$, которое удовлетворяет следующему условию:

$\forall\ \mathbf{T}_{\text{LL}} \in \mathbf{LLTraces}^{\text{fin}}\ \forall\ s_1 \in \mathbf{newtrace}(\mathbf{T}_{\text{LL}}, \alpha_1)\ \exists\ s_2 \in \mathbf{states}(\alpha_2(\mathbf{T}_{\text{LL}}))\colon (s_1, s_2) \in \mathbf{R}$.

Данное условие требует, чтобы при любом попадании в состояние абстрактной модели в отношении присутствовало соответствующее ему состояние детальной модели, в которое

детальная модель попала одновременно или ранее, чем случился данный переход в абстрактной модели.

Используя введённые термины, сформулируем следующий метод доказательства соответствия между разноуровневыми моделями в предположении, что верификация моделей **LTS1** и **LTS2** уже выполнена.

***Шаг* 1.** Определяется **R** – отношение соответствия между состояниями моделей **LTS1** и **LTS2**. Это можно сделать, используя термины только моделей **LTS1** и **LTS2**, то есть в виде машиночитаемой спецификации.

***Шаг* 2.** Заданное отношение **R** умозрительно проверяется на выполнение условие из определения 7. Проверка не может быть автоматизирована без построения модели событий низкого уровня.

***Шаг* 3**. Используя инструментальные средства автоматического или интерактивного доказательства доказывается утверждение:

$\forall\,(s_1, s_2) \in \mathbf{R}\; s_1 \in S_{\text{unsafe1}} \;\Rightarrow\; s_2 \in S_{\text{unsafe2}}$.

Если все шаги выполнены успешно, то можно сделать вывод, что целевые системы, описываемые моделью **LTS2**, удовлетворяют свойствам безопасности, доказанным для модели **LTS1**, т.е. $\mathbf{LLSafeT}_{\text{LTS2}} \subseteq \mathbf{LLSafeS}_{\text{LTS1}}$.

Доказательство.

Рассмотрим трассу $\mathbf{T}_{\text{LL}} \in \mathbf{LLSafeT}_{\text{LTS2}}$ и покажем, что тогда она принадлежит и $\mathbf{LLSafeS}_{\text{LTS1}}$. Предположим обратное, тогда найдётся $\mathbf{T'}_{\text{LL}} \in \mathbf{Prefix}^{\text{fin}}(\mathbf{T}_{\text{LL}})$, такой что $\mathbf{states}(\alpha_1(\mathbf{T'}_{\text{LL}})) \cap S_{\text{unsafe1}} \neq \emptyset$.

Обозначим s_k первое состояние в трассе $\alpha_1(\mathbf{T'}_{\text{LL}})$, такое что $s_k \in S_{\text{unsafe1}}$. Отметим, что s_k не может быть первым состоянием в трассе, т.к. тогда невозможно выполнить верификацию LTS1.

В силу конечности трассы $\mathbf{T'}_{\text{LL}}$ найдётся такой $\mathbf{T''}_{\text{LL}} \in \mathbf{Prefix}(\mathbf{T'}_{\text{LL}})$, что $s_k \in \mathbf{newstates}(\mathbf{T''}_{\text{LL}}, \alpha_1)$.

Тогда вследствие свойств отношения **R** найдётся $s_j \in \mathbf{states}(\mathbf{T''}_{\text{LL}}, \alpha_2)$: $(s_k, s_j) \in \mathbf{R}$. И так как $s_k \in S_{\text{unsafe1}}$, то по утверждению, сформулированному в Шаге 3, $s_j \in S_{\text{unsafe2}}$, а следовательно, найден $\mathbf{T''}_{\text{LL}} \in \mathbf{Prefix}^{\text{fin}}(\mathbf{T}_{\text{LL}})$, такой что $\mathbf{states}(\alpha_2(\mathbf{T''}_{\text{LL}})) \cap S_{\text{unsafe2}} \neq \emptyset$. Значит, $\mathbf{T}_{\text{LL}} \notin \mathbf{LLSafeS}_{\text{LTS2}}$, и т. к. верификация **LTS2** уже выполнена, то $\mathbf{T}_{\text{LL}} \notin \mathbf{LLSafeT}_{\text{LTS2}}$, что противоречит исходному предположению. □

3.2 Доказательство соответствия абстрактной модели поведения

Доказательства того, что целевые системы, описываемые более детальной моделью, удовлетворяют свойствам безопасности, доказанным для абстрактной модели не всегда достаточно. Некоторые значимые свойства могут быть отражены в абстрактной модели поведения, но не быть представлены в её свойствах безопасности. Тогда может быть поставлена задача доказательства того, что целевые системы, описываемые более детальной моделью, также соответствовали описанию корректного поведения абстрактной модели. Формально это можно сформулировать так: Если целевая система удовлетворяет требованию $\mathbf{LLSafeT}_{\text{LTS2}}$, то она удовлетворяет и требованию $\mathbf{LLSafeT}_{\text{LTS1}}$, или $\mathbf{LLSafeT}_{\text{LTS2}} \subseteq \mathbf{LLSafeT}_{\text{LTS1}}$.

В качестве первого подхода к решению данной задачи заметим, что если $\mathbf{LLSafeT}_{\text{LTS1}} = \mathbf{LLSafeS}_{\text{LTS1}}$, то из доказательства $\mathbf{LLSafeT}_{\text{LTS2}} \subseteq \mathbf{LLSafeS}_{\text{LTS1}}$ автоматически следует $\mathbf{LLSafeT}_{\text{LTS2}} \subseteq \mathbf{LLSafeT}_{\text{LTS1}}$.

Для выполнения равенства $\mathbf{LLSafeT}_{\text{LTS1}} = \mathbf{LLSafeS}_{\text{LTS1}}$ достаточно, чтобы

$\forall\,(s_1, \lambda_1, s'_1) \in \rightarrow_{\text{unsafe1}}\;\; s_1 \in S_{unsafe1} \vee s'_1 \in S_{unsafe1}$.

Если любой некорректный переход начинается в небезопасном состоянии или ведёт в небезопасное состояние, то любая трасса, не попадающая в $\mathbf{LLSafeT}_{\mathrm{LTS1}}$, также не попадёт в $\mathbf{LLSafeS}_{\mathrm{LTS1}}$. Поэтому если можно проверить модель **LTS1** на наличие этого свойства и если оно выполнено, то все свойства модели поведения отражены в её свойствах безопасности и дополнительных действий не требуется. Если это не так, то одним из способов решения поставленной задачи является доработка модели **LTS1**, чтобы это свойство стало выполняться.

Определение 8. Отношением соответствия между некорректными переходами моделей **LTS1** и **LTS2** будем называть отношение $\mathbf{RT} \subseteq \rightarrow_{\mathrm{unsafe1}} \times \rightarrow_{\mathrm{unsafe2}}$, которое удовлетворяет следующему условию:

$\forall\, \mathbf{T}_{\mathrm{LL}} \in \mathbf{LLTraces}^{fin}\ \forall\, (s_1, \lambda_1, s'_1) \in \mathbf{newtrace}(\mathbf{T}_{\mathrm{LL}}, \alpha_1) \cap \rightarrow_{\mathrm{unsafe1}}$

$s_1 \in S_{\mathrm{safe1}} \wedge s'_1 \in S_{\mathrm{safe1}} \Rightarrow \exists\, (s_2, \lambda_2, s'_2) \in \alpha_2(\mathbf{T}_{\mathrm{LL}})\colon \big((s_1, \lambda_1, s'_1), (s_2, \lambda_2, s'_2)\big) \in \mathbf{RT}$.

Данное условие требует, чтобы для любого некорректного перехода в абстрактной модели, начинающегося и завершающегося в безопасном состоянии, в отношении присутствовал соответствующий ему переход в детальной модели, который происходил одновременно или ранее, чем случился данный переход в абстрактной модели. Заметим, что когда $\forall (s_1, \lambda_1, s'_1) \in \rightarrow_{\mathrm{unsafe1}}\ s'_1 \in S_{\mathrm{unsafe1}}$, тогда отношение RT может быть вырожденным (пустым).

Сформулируем второй метод доказательства соответствия между разноуровневыми моделями в предположении, что верификация моделей **LTS1** и **LTS2** уже выполнена, а также при помощи первого метода установлено соотношение $\mathbf{LLSafeT}_{\mathrm{LTS2}} \subseteq \mathbf{LLSafeS}_{\mathrm{LTS1}}$.

***Шаг* 1**. Определяется **RT** отношение соответствия между некорректными переходами моделей **LTS1** и **LTS2**. Это можно сделать, используя термины только моделей **LTS1** и **LTS2**, то есть в виде машиночитаемой спецификации.

***Шаг* 2.** Заданное отношение **RT** умозрительно проверяется на выполнение условие из определения 8. Проверка не может быть автоматизирована без построения модели событий низкого уровня.

***Шаг* 3**. Используя инструментальные средства автоматического или интерактивного доказательства доказывается утверждение:

$\forall \big((s_1, \lambda_1, s'_1), (s_2, \lambda_2, s'_2)\big) \in \mathbf{RT}\ (s_1, \lambda_1, s'_1) \in \rightarrow_{\mathrm{unsafe1}} \wedge\ s_1 \in S_{\mathrm{safe1}} \wedge\ s'_1 \in S_{\mathrm{safe1}}$

$\Rightarrow\ (s_2, \lambda_2, s'_2) \in \rightarrow_{\mathrm{unsafe2}} \vee\ s_2 \in S_{\mathrm{unsafe2}} \vee s'_2 \in S_{\mathrm{unsafe2}}$.

Если все шаги выполнены успешно, то можно сделать вывод, что целевые системы, описываемые моделью **LTS2**, также соответствуют описанию корректного поведения, заданного абстрактной моделью **LTS1**, т.е. $\mathbf{LLSafeT}_{\mathrm{LTS2}} \subseteq \mathbf{LLSafeT}_{\mathrm{LTS1}}$.

Доказательство.

Рассмотрим трассу $\mathbf{T}_{\mathrm{LL}} \in \mathbf{LLSafeT}_{\mathrm{LTS2}}$ и покажем, что тогда она принадлежит и $\mathbf{LLSafeT}_{\mathrm{LTS1}}$. Предположим обратное, тогда найдётся $\mathbf{T'}_{\mathrm{LL}} \in \mathbf{Prefix}^{\mathrm{fin}}(\mathbf{T}_{\mathrm{LL}})$, такой что $\alpha_1(\mathbf{T'}_{\mathrm{LL}}) \cap \rightarrow_{\mathrm{unsafe1}} \neq \emptyset$.

Обозначим (s_k, λ_k, s'_k) первый переход в трассе $\alpha_1(\mathbf{T'}_{\mathrm{LL}})$, такой что $(s_k, \lambda_k, s'_k) \in \rightarrow_{\mathrm{unsafe1}}$.

Если $s_k \in S_{\mathrm{unsafe1}} \vee s'_k \in S_{\mathrm{unsafe1}}$, то $\mathbf{T'}_{\mathrm{LL}} \notin \mathbf{LLSafeS}_{\mathrm{LTS1}}$, а следовательно, $\mathbf{T'}_{\mathrm{LL}} \notin \mathbf{LLSafeT}_{\mathrm{LTS2}}$, и, соответственно, $\mathbf{T}_{\mathrm{LL}} \notin \mathbf{LLSafeT}_{\mathrm{LTS2}}$, что противоречит предположению.

В силу конечности трассы $\mathbf{T'}_{\mathrm{LL}}$ найдётся такой $\mathbf{T''}_{\mathrm{LL}} \in \mathbf{Prefix}(\mathbf{T'}_{\mathrm{LL}})$, что $(s_k, \lambda_k, s'_k) \in \mathbf{newtrace}(\mathbf{T}_{\mathrm{LL}}, \alpha_1)$.

Тогда вследствие свойств отношения **RT** найдётся $(s_j, \lambda_j, s'_j) \in \alpha_2(\mathbf{T''}_{\mathrm{LL}})\colon ((s_k, \lambda_k, s'_k), (s_j, \lambda_j, s'_j)) \in \mathbf{RT}$. И так как $(s_k, \lambda_k, s'_k) \in \rightarrow_{\mathrm{unsafe1}} \wedge\ s_k \in S_{\mathrm{safe1}} \wedge\ s'_k \in S_{\mathrm{safe1}}$, то по утверждению, сформулированному в Шаге 3, $(s_2, \lambda_2, s'_2) \in \rightarrow_{\mathrm{unsafe2}} \vee\, s_2 \in S_{\mathrm{unsafe2}} \vee\ s'_2 \in S_{\mathrm{unsafe2}}$. А следовательно, найден $\mathbf{T''}_{\mathrm{LL}} \in \mathbf{Prefix}^{\mathrm{fin}}(\mathbf{T}_{\mathrm{LL}})$, такой что $\mathbf{states}(\alpha_2(\mathbf{T''}_{\mathrm{LL}})) \cap S_{\mathrm{unsafe2}} \neq \emptyset$ или $\alpha_2(\mathbf{T''}_{\mathrm{LL}}) \cap \rightarrow_{\mathrm{unsafe2}} \neq \emptyset$.

Значит $\mathbf{T}_{\mathrm{LL}} \notin \mathbf{LLSafeT}_{\mathrm{LTS2}}$, что противоречит исходному предположению. □

4. Вопросы практического применения методов

Относительно традиционных подходов, основанных на отношении уточнения, которые обеспечивают установления более сложных соответствий между разноуровневыми моделями, вплоть до отношения бисимуляции, предложенные методы доказательства соответствия между разноуровневыми моделями нацелены на доказательство свойств безопасности в индивидуальной трассовой семантике, что позволяет сделать их более простыми и пригодными для применения при решении практических задач верификации сложных видов функциональных требований, таких как ограничения на возникающие информационные потоки в операционных системах ответственного применения.

Например, при верификации моделей управления доступом для операционных систем, построенных на основе ядра Linux [12, 13, 14], возникает три уровня моделей, для которых применение методов на основе отношения уточнения, оказывается проблематичным. Наиболее абстрактный уровень [15] соответствует семейству требований доверия к безопасности ADV_SPM «Моделирование политики безопасности» в терминологии стандарта ГОСТ Р ИСО/МЭК 15408 [16-18]. В нём описываются вопросы управления доступом и информационных потоков в абстрактных терминах пользователей, субъектов, объектов и т. д., которые не всегда прямо соответствуют терминам, в которых работает операционная система.

Следующий уровень соответствует семейству требований доверия к безопасности ADV_FSP «Функциональная спецификация» ГОСТ Р ИСО/МЭК 15408. В нём описываются требования к поведение целевой системы в терминах её интерфейса. В случае с операционными системами это может быть интерфейс системных вызовов, в котором фигурируют понятия процессов, файловых дескрипторов, сокетов и т. д.

Ещё один уровень возникает при моделировании поведения отдельных компонентов целевой системы, непосредственно реализующих требования безопасности. Например, в случае операционных систем это может быть компонент ядра, реализующих интерфейс LSM (Linux Security Module), в котором операции выполняются над внутренними структурами ядра, представляющими файл (inode) или запись о файле в директории (dentry).

Соответственно, предложенные методы могут быть использованы для доказательства соответствия между всеми видами моделей SPM-FSP, FSP-LSM и SPM-LSM.

4.1 Направления дальнейших исследований

Практические проблемы, которые при этом возникают, связаны с двумя моментами. Во-первых, традиционные языки формальных спецификаций нацелены на описание корректных моделей поведения, то есть для описания **LTS** в составе $(S, S_{\mathrm{init}}, L, \rightarrow_{\mathrm{safe}}, S_{\mathrm{safe}})$. Вопросы адекватной спецификации некорректных поведений $\rightarrow_{\mathrm{unsafe}}$ требуют дополнительного исследования. Вторым моментом, требующим анализа на практике, являются умозрительные проверки, выполняемые в рамках некоторых шагов предлагаемых методов. В то же время следует отметить, что явные и чёткие формулировки таких проверок являются предпочтительными по сравнению с достаточно распространённым подходом оставлять вопросы адекватности моделей своим неформализуемым прообразам за рамками рассмотрения.

Рассматривая вопрос спецификации некорректных поведений $\rightarrow_{\mathrm{unsafe}}$, в первую очередь, следует пояснить необходимость расширения для этих целей множества меток. Рассмотрим, например, модель целевой системы, в которой вызов функции умножения на два **dmul** представлен в множестве меток **L** событиями dmul(0,0), dmul(1,2), dmul(2,4) и т. д. Тогда если реализация этой функции для параметра 2 вернула 5 в модели не окажется метки dmul(2,5) для представления этого события. Таким образом, одним из шагов по описанию

спецификации некорректных поведений является расширение множества меток для представления таких некорректных событий. Во многих случаях это можно сделать путём автоматической трансформации исходной спецификации по некоторым правилам. Например, за счёт допуска нарушения ограничений на параметры событий. Тем не менее, это не всегда возможно, и тогда альтернативным решением может стать моделирование некорректным событий выделенной меткой **fail**. Но следует иметь в виду, что введение таких неестественных (аномальных) трансформаций может существенно усложнить умозрительные проверки требуемых свойств и тем самым повысить вероятность пропуска ошибки в ходе такой проверки.

Относительно спецификации некорректных переходов может быть рассмотрено два крайних взгляда. Если при интерпретации модели **LTS** в виде функции α состояние модели рассматривать как прямое отображение состояния реализации (взгляд белого ящика), то естественным шагом будет дополнение корректных переходов некорректными из каждого состояния в каждое с любой возможной меткой. Если же состояние модели рассматривать исключительно как модельное, формируемое по результатам предыдущих событий (взгляд чёрного ящика), то модель необходимо расширять, допуская появление всех возможных меток, а вот переходы определять в соответствии с её внутренней логикой поведения.

Ещё одним направлением для анализа является возможность совмещения предлагаемых методов с подходами динамической верификации и, в частности, тестирования на основе моделей. С одной стороны, при тестировании требуется решение задачи вынесения вердикта о корректности наблюдаемого поведения целевой системы, которое при наличии абстрактной модели реализуется через сбор трассы наблюдаемых событий, формирование на её основе абстрактных событий и соотнесения их с моделью. Поэтому при тестировании также актуален вопрос подхода к формированию функции α. С другой стороны, интересной возможностью может быть реализация автоматического проведения умозрительных проверок в ходе тестирования на тех трассах, которые будут возникать в ходе выполнения тестов.

4.2 Частичное соответствие между моделями

На практике также встречаются ситуации, когда абстрактная модель описывает более богатый спектр поведений, чем более детальная модель. Тогда более детальная модель не может соответствовать более абстрактной в соответствии с теми определениями, которые были введены выше. Тем не менее, потребность в доказательстве частичного соответствия между разноуровневыми моделями всё равно существует.

Для решения такой задачи предлагается предварительная трансформация абстрактной модели путём вычленения той функциональности, которая уже описана в детальной модели и дальнейшего применения предложенных методов относительно трансформированной абстрактной модели.

5. Заключение

В настоящей работе предложены методы доказательства соответствия между разноуровневыми моделями, нацеленные на доказательство свойств безопасности в индивидуальной трассовой семантике. Эти методы более просты и пригодны для применения при решении практических задач верификации сложных видов функциональных требований по сравнению с традиционными подходами, основанными на установлении отношения уточнения между моделями.

Для предложенных методов явно и чётко сформулированы умозрительные проверки, которые требуется выполнять вручную, поскольку их невозможно автоматизировать без построения модели низкоуровневых событий. Таким образом, методы позволяют хотя бы

ограниченным способом осуществить проверку адекватности моделей своим прообразам в физическом мире.

В качестве основного направления для дальнейших исследований рассматривается исследование вопросов практического применения предложенных методов, в частности, подходов к адекватной спецификации некорректных поведений, интеграции методов с существующими средствами спецификации и верификации формальных моделей, а также комбинация подхода с методами тестирования на основе моделей, в частности для частичной автоматизации проведения умозрительных проверок на тех трассах, которые будут возникать в ходе выполнения тестов.

Список литературы / References

[1]. B. Alpern, F.B. Schneider. Defining liveness. Information Processing Letters, vol. 21, issue 4, 1985, pp. 181-185.

[2]. A. Khoroshilov. On formalization of operating systems behaviour verification. In Proc. of 11th International Conference on Computer Science and Information Technologies (CSIT-2017), 2017, pp. 168-172.

[3]. В.В. Кулямин, Н.В. Пакулин, О.Л. Петренко, А.А. Сортов, А.В. Хорошилов. Формализация требований на практике. Препринт no. 13, ИСП РАН, 2006 г., 70 стр. / V.V. Kulyamin, N.V. Pakulin, O. L. Petrenko, A.A. Sortov, A.V. Khoroshilov. Formalization of requirements in practice. Preprint no. 13, ISP RAS, 2006, 70 pp. (in Russian).

[4]. J. He, C.A.R. Hoare, J.W. Sanders. Data refinement refined. Lecture Notes in Computer Science, vol. 213, 1986, pp. 187-196.

[5]. J.R. Abrial, D. Cansell, D. Méry. Refinement and reachability in Event-B. Lecture Notes in Computer Science, vol. 3455, 2005, pp. 222-241.

[6]. R.J.R. Back. Refinement of parallel and reactive programs. In M. Broy (ed). Program Design Calculi. Springer, 1993, pp. 73-92.

[7]. M. Butler. An approach to the design of distributed systems with B AMN. Lecture Notes in Computer Science, vol. 1212, 1997, pp. 223-241.

[8]. L. Aceto. Action Refinement in Process Algebras. Cambridge University Press, 1992, 283 p.

[9]. J. Derrick, E.A. Boiten. Non-atomic refinement in Z. Lecture Notes in Computer Science, vol. 1708, 1999, pp. 1477-1496.

[10]. G. Schellhorn. ASM refinement and generalizations of forward simulation in data refinement: A comparison. Theoretical Computer Sciences, vol. 336, issues 2-3, 2005, pp. 403–436.

[11]. П.Н. Девянин, В.В. Кулямин, А.Л. Оружейников, А.К. Петренко, А.В. Хорошилов, И.В. Щепетков. Способ верификации формальной автоматной модели поведения программной системы. Патент на изобретение №2682003, 2019 г. / P.N. Devyanin, V.V. Kulyamin, A.L. Oruzheinikov, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. A method for verifying a formal automaton model of a software system's behavior. Patent for invention No. 2682003, 2019 (in Russian).

[12]. П.Н. Девянин, Д.В. Ефремов, В.В. Кулямин, А.К. Петренко, А.В. Хорошилов, И.В. Щепетков. Моделирование и верификация политик безопасности управления доступом в операционных системах. М., Горячая линия – Телеком, 2019 г., 214 стр. / P.N. Devyanin, D.V. Efremov, V.V. Kulyamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Modeling and verification of access control security policies in operating systems. M., Hotline – Telecom, 2019, 214 p. (in Russian).

[13]. П.Н. Девянин, В.В. Кулямин, А.К. Петренко, А.В. Хорошилов, И.В. Щепетков. Интеграция мандатного и ролевого управления доступом и мандатного контроля целостности в верифицированной иерархической модели безопасности операционной системы. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 7-26. DOI: 10.15514/ISPRAS–2020–32(1)–1 / P.N. Devyanin, V.V. Kuliamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Integrating RBAC, MIC, and MLS in Verified Hierarchical Security Model for Operating System. Programming and Computer Software, 2020, vol. 46, issue 7, 2020, pp. 443–453. DOI: 10.1134/S0361768820070026.

[14]. V. Kuliamin, A. Khoroshilov, D. Medvedev. Formal Modeling of Multi-Level Security and Integrity Control Implemented with SELinux. In Proc. of the 2019 International Conference on Actual Problems of Systems and Software Engineering (APSSE), 2019, pp. 131-136.

[15]. А.В. Хорошилов, И.В. Щепетков. ADV_SPM – Формальные модели политики безопасности на практике. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 43-56 / A.V. Khoroshilov, I.V. Shchepetkov.

ADV_SPM – Formal security policy models in practice. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 3, 2017. pp. 43-56 (in Russian). DOI: 10.15514/ISPRAS-2017-29(3)-4.

[16]. ГОСТ Р ИСО/МЭК 15408-1-2012 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель / ISO/IEC 15408-1:2012 Information technology - Security techniques - Evaluation criteria for IT security - Part 1: Introduction and general model.

[17]. ГОСТ Р ИСО/МЭК 15408-2-2013 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности / ISO/IEC 15408-2:2013 Information technology - Security techniques - Evaluation criteria for IT security - Part 2: Security functional requirements.

[18]. ГОСТ Р ИСО/МЭК 15408-3-2013 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности / ISO/IEC 15408-3:2013 Information technology - Security techniques - Evaluation criteria for IT security - Part 3: Security assurance requirements.

Информация об авторе / Information about the author

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

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

DOI: 10.15514/ISPRAS–2020–32(6)–3

Формальная верификация модели мандатного контроля целостности в операционной системе KasperskyOS

В.С. Буренков, ORCID: 0000-0002-0232-774X <Vladimir.Burenkov@kaspersky.com>
АО «Лаборатория Касперского»,
125212, Россия, Москва, Ленинградское шоссе, д. 39А, стр. 3

Аннотация. Модели мандатного контроля целостности в операционных системах, как правило, накладывают ограничения на доступы активных компонент системы к пассивным и представляют эти доступы непосредственно. Такое представление можно использовать в случае операционных систем с монолитной архитектурой, где части системы, обеспечивающие доступ к ресурсам, входят в доверенную вычислительную базу. Однако доказательство отсутствия ошибок в таких компонентах и, следовательно, соответствия такой модели реальной системе является чрезвычайно трудной задачей. KasperskyOS – операционная система, основанная на микроядре и позволяющая организовать доступ к ресурсам с помощью компонентов, не обязательно входящих в доверенную вычислительную базу. Для KasperskyOS была разработана и реализована модель мандатного контроля целостности, учитывающая наличие таких компонентов и возможность их некорректного функционирования. В данной статье представлен процесс формализации этой модели и автоматизированного доказательства свойств, обеспечиваемых моделью. Описана разработка модели на языке Event-B и отражен опыт ее верификации с использованием платформы Rodin.

Ключевые слова: мандатный контроль целостности; Event-B; операционная система; KasperskyOS

Для цитирования: Буренков В.С. Формальная верификация модели мандатного контроля целостности в операционной системе KasperskyOS. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 31-48. DOI: 10.15514/ISPRAS–2020–32(6)–3

Formal Verification of a Mandatory Integrity Control Model for the KasperskyOS Operating System

V.S. Burenkov, ORCID: 0000-0002-0232-774X <Vladimir.Burenkov@kaspersky.com>
Kaspersky Lab
125212, Russia, Moscow, Leningradskoe shosse, 39A, 3

Abstract. Models of mandatory integrity control in operating systems usually restrict accesses of active components of a system to passive ones and represent the accesses directly. This is suitable in case of monolithic operating systems whose components that provide access to resources are part of the trusted computing base. However, due to the sheer complexity of such components, it is extremely nontrivial to prove such a model to be adequate to the real system. KasperskyOS is a microkernel operating system that organizes access to resources via components that are not necessarily part of the trusted computing base. KasperskyOS implements a specifically developed mandatory integrity control model that takes such components into account. This paper formalizes the model and describes the process of automated proof of the model's properties. For formalization, we use the Event-B language. We clarify parts specific to Event-B to make our presentation accessible to professionals familiar with discrete mathematics but not necessarily with Event-B. We reflect on the proof experience obtained in the Rodin platform.

Keywords: mandatory integrity control; Event-B; operating system; KasperskyOS

For citation: Burenkov V.S. Formal Verification of a Mandatory Integrity Control Model for the KasperskyOS Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 31-48 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–3

1. Введение

Контроль целостности компьютерных систем является одним из фундаментальных вопросов, рассматриваемых при разработке безопасных систем. Для обеспечения целостности в операционных системах реализуются механизмы мандатного контроля целостности [1]. Классический путь разработки механизмов безопасности начинается с создания модели системы [2]. Существуют и официальные требования наличия формальной модели политики безопасности. Их предъявляет, например, ФСТЭК России [3].

В данной работе рассматривается формализация и верификация модели мандатного контроля целостности, разработанной для и реализованной в новой микроядерной операционной системе KasperskyOS. Модель изначально была описана на математическом и естественном языках [4].

В работе кратко отмечены архитектурные особенности операционной системы KasperskyOS, нашедшие свое отражение в модели мандатного контроля целостности. Детально представлена формализация модели на языке Event-B [5]. Модель сопровождена пояснениями, облегчающими ее понимание для широкого круга специалистов, знакомых с основами дискретной математики, но не обязательно имеющим опыт работы с языком Event-B. Описаны решения по моделированию различных компонентов и конструкций модели. Сформулированы свойства безопасности, обеспечиваемые моделью, и описан опыт их доказательства с использованием платформы Rodin [6].

2. Выбор метода и инструмента верификации

Для верификации моделей управления доступом используют как метод проверки модели [7–10], так и методы дедуктивной верификации [11–15]. Метод проверки моделей подвержен проблеме взрыва числа состояний. Поэтому его использование приведет либо к необходимости ограничения размерности множеств субъектов и объектов моделей (то есть к неполной верификации), либо к необходимости разработки модификаций метода (например, с целью получения абстрактных моделей) и специализированных инструментов.

Методы дедуктивной верификации не накладывают ограничений на размерность элементов модели, однако заведомо требуют ручного вмешательства. Тем не менее, например, язык Event-B, одной из основ которого является теория множеств, позволяет достаточно удобно описывать модели, состояние которых представлено множествами и отношениями, а изменения состояний – посредством бинарного отношения на множестве состояний, или, другими словами, событий или правил преобразования состояний. Именно в таких терминах описана модель мандатного контроля целостности для KasperskyOS в работе [4]. Использование же платформы Rodin, представляющей инфраструктуру для верификации моделей на языке Event-B, позволяет зачастую достичь высокой степени автоматизации процесса доказательства [16]. Ввиду этих причин в данной работе выбран язык Event-B для формализации модели мандатного контроля целостности.

3. Архитектура операционной системы KasperskyOS

Модель, формализуемая в данной работе, была разработана и реализована в микроядерной операционной системе KasperskyOS и отражает ряд архитектурных свойств этой системы. Основными компонентами операционной системы являются микроядро, монитор безопасности и множество драйверов, реализованных в виде непривилегированных

приложений. Главными функциями микроядра являются разделение ресурсов между процессами (которые в рамках операционной системы именуются *сущностями*), предоставление механизма межпроцессного взаимодействия IPC (interprocess communication) и запуск монитора безопасности для проверки этих взаимодействий. Операционная система разрабатывалась с учетом принципа разделения ресурсов и приложений посредством минимальной доверенной вычислительной базы, направленного на поддержку политик безопасности. Современным воплощением такого подхода является архитектура MILS [17].

Важным компонентом архитектуры MILS является *ядро разделения*, реализация которого гарантирует изоляцию приложений друг от друга. Изоляция приложений очень важна с точки зрения безопасности, так как она позволяет ограничить нежелательное воздействие на систему со стороны отдельных приложений.

Однако, во многих случаях одного только ядра разделения недостаточно: необходимо также убедиться, что взаимодействия в системе удовлетворяют требованиям безопасности. В KasperskyOS для решения этой проблемы существует Kaspersky Security System – набор инструментов для спецификации различных свойств безопасности и синтеза монитора обращений, проверяющего любые взаимодействия внутри системы на предмет соответствия политике. Для спецификации свойств (политики) используется декларативный язык Policy Specification Language (PSL). Мандатный контроль целостности представлен в языке множеством правил, соответствующих правилам модели мандатного контроля целостности. Описанная на PSL политика транслируется в запускаемый образ монитора безопасности, которому ядро и другие сервисы могут направлять свои запросы.

Каждая сущность или ресурс является *доменом безопасности*, с которым ассоциирован *контекст безопасности*, определяемый *идентификатором*. Таким образом, с точки зрения политики безопасности, программно-аппаратная система, управляемая KasperskyOS, представляет собой множество доменов безопасности и информационных потоков между ними.

3.1 Взаимодействие между процессами

В KasperskyOS сущности обмениваются друг с другом сообщениями посредством механизма синхронного межпроцессного взаимодействия. Выделяются две роли: клиент (сущность, инициирующая взаимодействие) и сервер (сущность, обрабатывающая обращение). Клиент направляет серверу сообщение-*запрос*, при этом поток исполнения, из которого производится обращение, блокируется до получения ответа от сервера или ядра (в случае, например, ошибки). Серверный поток исполнения находится в ожидании сообщений. При получении запроса, этот поток получает управление, обрабатывает запрос и отсылает сообщение-*ответ*. При получении сообщения-ответа клиентский поток может продолжать исполнение.

Монитор безопасности имеет возможность контролировать любые IPC-сообщения. С этой целью ядро передает монитору на проверку как сообщение-запрос, так и ответ. Передача сообщения происходит только в случае, если монитор дал разрешение. Кроме операций по контролю запросов и ответов, монитор также поддерживает две специальных операции – execute и security.

Операция *execute* используется ядром для сообщения монитору о создании новой сущности. Операция *security* позволяет сущностям отправлять сообщения монитору безопасности напрямую. Данная операция является основой для реализации сервисов (драйверов), которые предоставляют контролируемый доступ к ресурсам.

3.2 Управление доступом к ресурсам

В KasperskyOS большинство сервисов реализуется как непривилегированные приложения. Такие сервисы часто предоставляют доступ к ресурсам определенного вида. Смысл понятия

«ресурс» определяется самим сервисом. Например, для файловой системы ресурсами могут быть файлы и каталоги. Поскольку данные сервисы не входят в ядро операционной системы, то контролировать доступ к ним только возможностями ядра невозможно. В связи с этим для разграничения доступа к таким ресурсам в KasperskyOS выработан следующий поход.

Ключевым понятием является *драйвер* – это процесс-сервис, который вводит в систему новый тип ресурсов. Доступ к ресурсам определенного типа возможен только путем обращения к соответствующему драйверу. С каждым ресурсом драйвер ассоциирует системный дескриптор, с которым подсистема безопасности связывает информацию, необходимую для контроля доступа (например, метку целостности).

Во время каждого доступа к ресурсам драйвер ресурсов должен предоставлять монитору безопасности необходимый контекст. Монитор безопасности затем разрешает или запрещает доступ. При доступе к ресурсу, драйвер использует операцию security для обращения к подсистеме безопасности. В этом обращении драйвер передает дескриптор ресурса и любую дополнительную информацию, которую посчитает необходимой для осуществления контроля. Например, при чтении файла передается информация о том, кто (дескриптор клиента) хочет выполнить операцию (чтение файла), над чем (дескриптор файла). Монитор безопасности применяет ассоциированные с этим обращением правила и возвращает результат проверки. Ответственность драйвера – правильно проинтерпретировать ответ монитора: заблокировать операцию, если она не была разрешена. Таким образом, для реализации контроля над ресурсами, драйвер должен предоставить механизм, а политика определяется монитором безопасности. Такая архитектура позволяет подсистеме безопасности единообразно трактовать как системные ресурсы (в первую очередь, сущности и IPC-каналы), так и прочие ресурсы, введенные в систему драйверами. Ядро операционной системы при этом занимается только управлением (выдачей и квотированием) системных дескрипторов.

Ядро операционной системы гарантирует разделение адресных пространств сущностей, поэтому драйвер может управлять только теми ресурсами, системные дескрипторы на которые были выданы именно этому драйверу.

Безусловно, уровень доверия (в самом общем смысле) некоторого ресурса не может быть выше, чем уровень доверия драйвера этого ресурса.

4. Разработка и верификация формальной модели мандатного контроля целостности

4.1 Контекст модели

Контекст модели определяет ее неизменное состояние. Далее представлены множества, константы и аксиомы контекста. Имена аксиом предварены символом `@`.

4.1.1 Сущности и объекты

Сущности (субъекты) системы представляют активные компоненты операционной системы.

Объекты системы представляют пассивные компоненты операционной системы. Все сущности и объекты модели принадлежат множеству `Entities_and_objects_universe`:

```
@Entities_and_objects_universe_finite
    finite(Entities_and_objects_universe) // множество конечно
```

Среди сущностей выделяется сущность `core`, представляющая ядро операционной системы KasperskyOS:

```
@core_type
    core ∈ Entities_and_objects_universe
```

4.1.2 Доступы и информационные потоки

Множество *видов доступа* `Access_types` состоит из двух констант `read_a` и `write_a`, где `read_a` обозначает доступ на чтение, `write_a` обозначает доступ на запись:

```
@Access_types
    partition(Access_types, {read_a}, {write_a})
```

Множество *видов информационных потоков* `Flow_types` включает в себя константу `write_m`, обозначающую информационный поток по памяти на запись в сущность или объект:

```
@Flow_types
    Flow_types = {write_m}
```

4.1.3 Уровни целостности

Уровни целостности представлены в модели конечным множеством `Integrity_levels`, на котором определено отношение частичного порядка `le_il` («≤»):

```
@Integrity_levels_finite
    finite(Integrity_levels)
@le_il_type
    le_il ⊆ Integrity_levels × Integrity_levels
@le_il_reflexive // le_il является рефлексивным
    ∀x · x ∈ Integrity_levels ⇒ x ↦ x ∈ le_il
@le_il_transitive // le_il является транзитивным
    ∀x, y, z · x ∈ Integrity_levels ∧ y ∈ Integrity_levels ∧
                z ∈ Integrity_levels ∧
      x ↦ y ∈ le_il ∧ y ↦ z ∈ le_il ⇒ x ↦ z ∈ le_il
@le_il_antisymmetric // le_il является антисимметричным
    ∀x, y · x ∈ Integrity_levels ∧ y ∈ Integrity_levels ∧
      x ↦ y ∈ le_il ∧ y ↦ x ∈ le_il ⇒ x = y
```

На основе отношения `le_il` также определено отношение `l_il` («<»):

```
@l_il_type
    l_il ⊆ Integrity_levels × Integrity_levels
@l_il_def
    ∀x, y · x ∈ Integrity_levels ∧ y ∈ Integrity_levels ∧
      x ↦ y ∈ l_il ⇔ x ↦ y ∈ le_il ∧ x ≠ y
```

Заметим, что в качестве множества уровней целостности в моделях мандатного контроля целостности обычно используют решетку. Превратить множество `Integrity_levels` в решетку можно добавлением аксиом существования наибольшей нижней грани (инфимума) `inf_il` и наименьшей верхней грани (супремума) `sup_il`. Например:

```
@inf_exists
    inf_il ∈ Integrity_levels ∧
      (∀x · x ∈ Integrity_levels ⇒ inf_il ↦ x ∈ le_il) ∧
      (∀lb · lb ∈ Integrity_levels ∧
       (∀y · y ∈ Integrity_levels ⇒ lb ↦ y ∈ le_il) ⇒ lb ↦ inf_il ∈ le_il)
```

4.2 Машина модели

Машина модели содержит ее динамическую часть. Далее представлены переменные, инварианты и события, определяющие машину. Имена инвариантов (свойств, которые всегда должны выполняться), предварены символом `@`.

4.2.1 Сущности и объекты

`Entities` – множество сущностей системы, среди которых выделяется особая сущность – ядро операционной системы:

```
@Entities_type
    Entities ⊆ Entities_and_objects_universe
@core_type
    core ∈ Entities
```

`Objects` – множество объектов системы:

```
@Objects_type
    Objects ⊆ Entities_and_objects_universe
```

Сущности и объекты представлены разными множествами, поскольку они имеют различную природу. Однако в определении некоторых элементов модели (например, множества информационных потоков) используются как сущности, так и объекты. Чтобы в таких случаях иметь возможность использовать объединение множеств сущностей и объектов, эти множества должны быть подмножествами одного универсума, определенного в контексте модели. Заметим, что множества сущностей и объектов не пересекаются:

```
@Entities_and_objects_differ
    Entities ∩ Objects = ∅
```

Для целей моделирования иерархии объектов введена *функция иерархии объектов* `Container_contents`, которая ставит в соответствие каждому объекту множество объектов:

```
@Container_contents_type
    Container_contents ∈ Objects → ℙ(Objects)
```

Будем говорить, что объекты из `Container_contents(c)` непосредственно находятся в контейнере `c`.

Выделяется множество корневых объектов `Root_objects`, не находящихся ни в каком контейнере:

```
@Root_objects_type
    Root_objects ⊆ Objects
```

Для целей моделирования управления работой с ресурсами системы посредством драйверов ресурсов, в модели введено понятия *драйвера объектов*. Драйвер объектов – сущность, посредством которой выполняются операции с объектами. Будем говорить, что драйвер объектов управляет подмножеством объектов.

Функция `Resource_driver` ставит в соответствие любому объекту системы его драйвер:

```
@Resource_driver_type
    Resource_driver ∈ Objects → Entities
```

Несмотря на то что драйверы ресурсов и другие прикладные сущности не обязательно входят в доверенную вычислительную базу, их корректное функционирование предполагает удовлетворение определенных требований [4]. Однако по каким-либо причинам (например, ошибка в исходном коде сущности) эти требования могут быть не выполнены. Описание таких причин может быть составлено на основе деталей реализации системы, что находится за рамками модели. В то же время модель позволяет выполнить анализ последствий нарушения требований и возникновения непредвиденных событий в системе. Для этого в модели введено понятие захвата контроля над сущностью или объектом. Если над сущностью захвачен контроль, то предположения о ее поведении более не выполняются. Захват контроля над объектом означает появление в нем данных, которые он не должен содержать при корректном функционировании системы. Формально последствия захвата контроля над сущностями и объектами описаны далее в правилах преобразования состояний.

Множества `Entities_compromised` и `Objects_compromised` определяют сущности и объекты, над которыми захвачен контроль:

```
@Entities_compromised_type
    Entities_compromised ⊆ Entities
@Objects_compromised_type
    Objects_compromised ⊆ Objects
```

Как будет доказано в подразделе 4.3, модель исключает неограниченный захват контроля частей системы: если захвачен контроль над частью системы, максимальный уровень целостности компонент которой ниже максимально возможного уровня целостности, то это не приводит к захвату контроля над более целостными компонентами системы и, как следствие, всей системы в целом.

4.2.2 Доступы и информационные потоки

`Accesses` – множество *доступов* сущностей к объектам:

```
@Accesses_type
    Accesses ⊆ Entities × Objects × Access_types
```

`Flows` – множество *информационных потоков*:

```
@Flows_type
    Flows ⊆ (Entities ∪ Objects) × (Entities ∪ Objects) × Flow_types
```

4.2.3 Уровни целостности

Уровень целостности сущностей и объектов определен с помощью функции `integrity_level`:

```
@integrity_level_type
    integrity_level ∈ (Entities ∪ Objects) → Integrity_levels
```

При работе операционной системы возникают ситуации, в которых возможно участие сущности в «опасных» взаимодействиях, то есть обращениях на чтение к объектам с более низким либо несравнимым уровнем целостности. Для разрешения таких взаимодействий в модели введена функция `integrity_level_r`, определяющая для каждой сущности минимальный уровень целостности объектов, к которым эта сущность может обращаться на чтение:

```
@integrity_level_r_type
    integrity_level_r ∈ Entities → Integrity_levels
```

В модели также введена *функция привилегий* `upgrade_privilege`, служащая для обозначения сущностей, которым разрешено повышать уровень целостности объектов:

```
@upgrade_privilege_type
    upgrade_privilege ∈ Entities → BOOL
```

Предполагается, что в качестве таких сущностей могут выступать доверенные сущности, для которых значение данной функции устанавливается равным `TRUE`. Заметим, что в рамках модели мы не рассматриваем процесс определения функции привилегий. Предполагается, что ее значение всегда определено корректно. Если ее значение равно `TRUE` для некоторой сущности `s`, то эта сущность действительно обладает привилегией повышения уровня целостности объектов. Если же сущность `s` не обладает такой привилегией, то значение функции привилегий для нее равно `FALSE`.

4.2.4 События модели

События разработаны в соответствии с правилами преобразования состояний, определенными в работе [4]. Все события, кроме события инициализации, включают защиты

– логические условия, определяющие, в каких случаях событие может быть исполнено, – и действия, описывающие результат исполнения события. Событие инициализации включает только действия. Имена защит в представленной модели начинаются с `@grd`, а имена действий – с `@act`.

Инициализация начального состояния модели

Событие инициализации описывает начальное состояние модели. Модель разрабатывалась таким образом, чтобы к ее начальному состоянию предъявлялось как можно меньше требований. Предполагается, что в начальном состоянии существует сущность `core`, и определены значения ее уровней целостности с учетом того, что ядро операционной системы является наиболее доверенной сущностью.

```
event INITIALISATION
    then
      @act1 Entities ≔ {core}
      @act2 Objects ≔ ∅
      @act3 Entities_compromised ≔ ∅
      @act4 Objects_compromised ≔ ∅
      @act5 Accesses ≔ ∅
      @act6 Flows ≔ ∅
      @act7 integrity_level ≔ {core ↦ sup_il}
      @act8 integrity_level_r ≔ {core ↦ sup_il}
      @act9 upgrade_privilege ≔ {core ↦ FALSE}
      @act10 Root_objects ≔ ∅
      @act11 Container_contents ≔ ∅ × {∅}
      @act12 Resource_driver ≔ ∅
end
```

Получение доступа на чтение к объектам

Событие `access_read` определяет получение сущностью `entity` доступа на чтение к объекту `object` посредством драйвера `driver`. Возникающие в результате применения правила информационные потоки зависят от того, является ли драйвер скомпрометированным. Множество информационных потоков формируется в действии `act_basic2` события. Для определения потоков в зависимости от факта компрометации драйвера используются конструкции *set comprehension*, частью условия которых является принадлежность драйвера множеству скомпрометированных сущностей. Такой подход позволяет моделировать конструкции *if-then-else*, используемые в модели на математическом и естественном языках.

```
event access_read
  any
    entity
    driver
    object
  where
    @grd_basic1 entity ∈ Entities
    @grd_basic2 driver ∈ Entities
    @grd_basic3 object ∈ Objects
    @grd_basic4 Resource_driver(object) = driver
    @grd_mic1 (integrity_level(entity) ↦ integrity_level(driver) ∈ le_il)∨
        (¬(integrity_level(entity) ↦ integrity_level(driver) ∈ le_il)
```

```
          ∧ integrity_level_r(entity) ↦ integrity_level(driver) ∈ le_il)
    @grd_mic2 driver ∉ Entities_compromised ⇒
        ((integrity_level(entity) ↦ integrity_level(object) ∈ le_il) ∨
          (¬(integrity_level(entity) ↦ integrity_level(object) ∈ le_il)
          ∧ integrity_level_r(entity) ↦ integrity_level(object) ∈ le_il))
    @grd_mic3 integrity_level(object) ↦ integrity_level(driver) ∈ le_il
  then
    @act_basic1 Accesses ≔ Accesses ∪ {entity ↦ object ↦ read_a}
    @act_basic2 Flows ≔ Flows ∪
        {o ↦ e ↦ w ∣ o = object ∧ e = entity ∧ w = write_m ∧
                     driver ∉ Entities_compromised ∧
         ((integrity_level(entity) ↦ integrity_level(object) ∈ le_il ∧
           integrity_level(entity) ↦ integrity_level(driver) ∈ le_il) ∨
                       entity ∈ Entities_compromised)} ∪
        {o ↦ e ↦ w ∣ o = object ∧ e = entity ∧ w = write_m ∧
                  driver ∈ Entities_compromised ∧
         (integrity_level(entity) ↦ integrity_level(driver) ∈ le_il ∨
                       entity ∈ Entities_compromised)} ∪
        {d ↦ e ↦ w ∣ d = driver ∧ e = entity ∧ w = write_m ∧
                  driver ∈ Entities_compromised ∧
         (integrity_level(entity) ↦ integrity_level(driver) ∈ le_il ∨
                           entity ∈ Entities_compromised)}
end
```

Получение доступа на запись к объектам

Событие `access_write` определяет получение сущностью `entity` доступа на запись к объекту `object` посредством драйвера `driver`. Возникающие информационные потоки для этого события описываются проще, чем в случае получения доступа на чтение, поэтому в действии act_basic2 вместо оператора присваивания и конструкций set comprehension использована формула, устанавливающая связь между переменными после (со штрихом) и до (без штриха) выполнения события. Такой подход еще более близок к модели на математическом и естественном языках.

```
event access_write
  any
    entity
    driver
    object
  where
    @grd_basic1 entity ∈ Entities
    @grd_basic2 driver ∈ Entities
    @grd_basic3 object ∈ Objects
    @grd_basic4 Resource_driver(object) = driver
    @grd_mic1 driver ∉ Entities_compromised ⇒
                integrity_level(object) ↦ integrity_level(entity) ∈ le_il
    @grd_mic2 integrity_level(object) ↦ integrity_level(driver) ∈ le_il
  then
    @act_basic1 Accesses ≔ Accesses ∪ {entity ↦ object ↦ write_a}
    @act_basic2 Flows :| (Flows' = Flows ∪
```

```
        {entity ↦ object ↦ write_m} ∧ driver ∉ Entities_compromised) ∨
                        (Flows' = Flows ∪ {entity ↦ object ↦ write_m,
entity ↦ driver ↦ write_m} ∧ driver ∈ Entities_compromised)
end
```

Создание сущностей и объектов, удаление, перемещение, повышение уровня целостности объектов

Событие `execute` определяет, как сущность `creator` создает (запускает) сущность `new_entity` из объекта `image` с установлением уровней целостности сущности `new_entity`.

```
event execute
  any
    creator
    image
    new_entity
    il
    ilr
  where
    @grd_basic1 creator ∈ Entities
    @grd_basic2 image ∈ Objects
    @grd_basic3 new_entity ∈ Entities_and_objects_universe
    @grd_basic4 new_entity ∉ Entities ∪ Objects
    @grd_mic1 il ∈ Integrity_levels
    @grd_mic2 ilr ∈ Integrity_levels
    @grd_mic3 il ↦ integrity_level(image) ∈ le_il
    @grd_mic4 ilr ↦ il ∈ le_il
  then
    @act_basic1 Entities ≔ Entities ∪ {new_entity}
    @act_basic2 Flows :| (Flows' = Flows ∪
          {image ↦ new_entity ↦ write_m} ∧ image ∈ Objects_compromised) ∨
                        (Flows' = Flows ∧ image ∉ Objects_compromised)
    @act_basic3 upgrade_privilege(new_entity) ≔ FALSE
    @act_mic1 integrity_level(new_entity) ≔ il
    @act_mic2 integrity_level_r(new_entity) ≔ ilr
end
```

Событие `create_object` определяет, как сущность `creator` создает объект `new_object` посредством драйвера `driver`, включает `new_object` в состав контейнера `container` и устанавливает уровень целостности `new_object` в `il`. Для описания изменений в функции иерархии объектов: модификации ее значения для объекта `container` и добавления в область определения нового объекта `new_entity` с установлением значения функции для него в ∅ использован оператор relational overriding ∃.

```
event create_object
  any
    creator
    new_object
    container
    driver
    il
  where
    @grd_basic1 creator ∈ Entities
```

```
    @grd_basic2 new_object ∈ Entities_and_objects_universe
    @grd_basic3 new_object ∉ Entities ∪ Objects
    @grd_basic4 container ∈ Objects
    @grd_basic5 driver ∈ Entities
    @grd_basic6 creator ↦ container ↦ write_a ∈ Accesses
    @grd_basic7 driver ↦ container ↦ write_a ∈ Accesses
    @grd_mic1 il ∈ Integrity_levels
    @grd_mic2 il ↦ integrity_level(creator) ∈ le_il
    @grd_mic3 il ↦ integrity_level(container) ∈ le_il
    @grd_mic4 il ↦ integrity_level(driver) ∈ le_il
  then
    @act_basic1 Objects ≔ Objects ∪ {new_object}
    @act_basic2 Container_contents ≔ Container_contents ∃
      {container ↦ (Container_contents(container) ∪{new_object}),
       new_object ↦ ∅}
    @act_basic_drr Resource_driver(new_object) ≔ driver
    @act_basic3 Flows :| (Flows' = Flows ∪
      {creator ↦ new_object ↦ write_m, creator ↦ driver ↦ write_m}
                                    ∧ driver ∈ Entities_compromised) ∨
                        (Flows' = Flows ∧ driver ∉ Entities_compromised)
    @act_basic4 Objects_compromised :| (driver ∈ Entities_compromised ∧
       Objects_compromised' = Objects_compromised ∪ {new_object} ) ∨
                                        (driver ∉ Entities_compromised ∧
       Objects_compromised' = Objects_compromised)
    @act_mic1 integrity_level(new_object) ≔ il
end
```

Событие `create_root` по созданию корневых объектов определено аналогично событию `create_object`.

Событие `move_object` определяет, как сущность `entity` перемещает объект `object` из контейнера `from` в контейнер `to` посредством драйвера `driver`.

```
event move_object
  any
    entity
    object
    driver
    from
    to
  where
    @grd_basic1 entity ∈ Entities
    @grd_basic2 object ∈ Objects
    @grd_basic3 driver ∈ Entities
    @grd_basic4 Resource_driver(object) = driver
    @grd_basic5 from ∈ Objects
    @grd_basic6 to ∈ Objects
    @grd_basic61 from ≠ to
    @grd_basic7 object ∈ Container_contents(from)
    @grd_basic8 entity ↦ from ↦ write_a ∈ Accesses
```

```
    @grd_basic9 entity ↦ to ↦ write_a ∈ Accesses
    @grd_basic10 driver ↦ from ↦ write_a ∈ Accesses
    @grd_basic11 driver ↦ to ↦ write_a ∈ Accesses
    @grd_basic12 object ≠ from
    @grd_basic13 object ≠ to
    @grd_mic1 driver ∉ Entities_compromised ⇒
         integrity_level(object) ↦ integrity_level(entity) ∈ le_il
    @grd_mic2 integrity_level(object) ↦ integrity_level(driver) ∈ le_il
    @grd_mic3 integrity_level(object) ↦ integrity_level(to) ∈ le_il
  then
    @act_basic1 Container_contents ≔ Container_contents ∃
            {from ↦ (Container_contents(from) ∖ {object}),
             to ↦ (Container_contents(to) ∪ {object})}
    @act_basic2 Flows :| (driver ∈ Entities_compromised ∧ Flows' = Flows ∪
       {entity ↦ object ↦ write_m, entity ↦ driver ↦ write_m}) ∨
                          (driver ∉ Entities_compromised ∧ Flows' = Flows)
end
```

Событие `delete_object` определяет, как сущность `entity` удаляет объект `object` из контейнера `container` посредством драйвера `driver`. Для удаления объекта `object` из области определения функций `Container_contents`, `integrity_level` и `Resource_driver` используется оператор domain subtraction ⩤.

```
event delete_object
  any
    entity
    object
    driver
    container
  where
    @grd_basic1 entity ∈ Entities
    @grd_basic2 object ∈ Objects
    @grd_basic21 driver ∈ Entities
    @grd_basic3 container ∈ Objects
    @grd_basic4 object ∈ Container_contents(container)
    @grd_basic41 Resource_driver(object) = driver
    @grd_basic5 entity ↦ container ↦ write_a ∈ Accesses
    @grd_basic51 driver ↦ container ↦ write_a ∈ Accesses
    @grd_basic6 Container_contents(object) = ∅
    @grd_mic1 integrity_level(object) ↦ integrity_level(entity) ∈ le_il
    @grd_mic2 integrity_level(object) ↦ integrity_level(driver) ∈ le_il
  then
      @act_basic1 Objects ≔ Objects ∖ {object}
      @act_basic2 Objects_compromised :| (object ∈ Objects_compromised ∧
            Objects_compromised' = Objects_compromised ∖ {object}) ∨
                                        (object ∉ Objects_compromised ∧
            Objects_compromised' = Objects_compromised)
      @act_basic3 Accesses ≔ Accesses ∖
             {s ↦ o ↦ r | s ∈ Entities ∧ o = object ∧ r ∈ Access_types}
```

```
      @act_basic4 Flows ≔ (Flows ∪
              {e ↦ d ↦ w | e = entity ∧ d = driver ∧ d ∈ Entities_compromised
∧ w = write_m}) ∖ ({s ↦ o ↦ w | s ∈ Entities ∪ Objects ∧ o = object ∧ w =
write_m} ∪ {o ↦ s ↦ w | s ∈ Entities ∪ Objects ∧ o = object ∧ w = write_m})
      @act_basic5 Container_contents ≔ ({object} ⩤ Container_contents) ∃
           {container ↦ (Container_contents(container)∖{object})}
      @act_basic6 integrity_level ≔ {object} ⩤ integrity_level
      @act_basic7 Resource_driver ≔ {object} ⩤ Resource_driver
end
```

Событие `upgrade` определяет, как сущность `entity` изменяет уровень целостности объекта `object`, находящегося в контейнере `container`, посредством драйвера `driver`, на новое значение `il`. Операция `upgrade` должна выполняться в реальной системе только доверенными сущностями и подразумевает отсутствие информационных потоков к объекту, уровень целостности которого предполагается повысить. В противном случае источник такого информационного потока может стать менее целостным, чем приемник. В реальной системе для гарантии отсутствия таких потоков используются специальные средства (например, криптографические). Такие средства не рассматриваются в рамках модели. Поэтому при анализе модели на предмет свойств безопасности, которые она обеспечивает, предполагается, что доверенные сущности не выполняют операции, которые могут нарушить целостность системы, в частности, операцию `upgrade`. В модели это предположение отражено добавлением условия ⊥ в качестве защиты события `upgrade`. В отличие от полного исключения данного события из модели, такой подход позволяет выполнить анализ корректности определения защит и действий событий.

```
event upgrade
  any
    entity
    object
    container
    driver
    il
  where
    @grd_basic1 entity ∈ Entities
    @grd_basic2 object ∈ Objects
    @grd_basic3 container ∈ Objects
    @grd_basic4 Resource_driver(object) = driver
    @grd_basic5 object ∈ Container_contents(container)
    @grd_basic6 upgrade_privilege(entity) = TRUE
    @grd_mic1 integrity_level(object) ↦ integrity_level(entity) ∈ le_il
    @grd_mic2 il ↦ integrity_level(entity) ∈ le_il
    @grd_mic3 il ↦ integrity_level(container) ∈ le_il
    @grd_mic4 integrity_level(object) ↦ il ∈ l_il
    @grd ⊥
  then
    @act_mic1 integrity_level(object) ≔ il
end
```

Взаимодействие сущностей

Событие `call` определяет вызов сущностью `caller` метода сущности `callee` с целью получения данных от сущности `callee`.

```
event call
  any
    caller
    callee
  where
    @grd_basic1 caller ∈ Entities
    @grd_basic2 callee ∈ Entities
    @grd_mic1 (integrity_level(caller) ↦ integrity_level(callee) ∈ le_il)∨
          (¬(integrity_level(caller) ↦ integrity_level(callee) ∈ le_il) ∧
           integrity_level_r(caller) ↦ integrity_level(callee) ∈ le_il)
  then
    @act_basic1 Flows :|
((integrity_level(caller) ↦ integrity_level(callee) ∈ le_il ∨ caller ∈
Entities_compromised) ∧ ((Flows' = Flows ∪ {callee ↦ caller ↦ write_m}
                          ∧ callee ∉ Entities_compromised) ∨
 (Flows' = Flows ∪ {callee ↦ caller ↦ write_m, caller ↦ callee ↦ write_m}
                          ∧ callee ∈ Entities_compromised))) ∨
(¬(integrity_level(caller) ↦ integrity_level(callee) ∈ le_il ∨ caller ∈
Entities_compromised) ∧ Flows' = Flows)
end
```

Событие `invoke` определяет вызов сущностью `invoker` метода сущности `invokee` с целью передачи данных сущности `invokee`.

```
event invoke
  any
    invoker
    invokee
  where
    @grd_basic1 invoker ∈ Entities
    @grd_basic2 invokee ∈ Entities
    @grd_mic1 integrity_level(invokee) ↦ integrity_level(invoker) ∈ le_il
  then
    @act_basic1 Flows :| (Flows' = Flows ∪ {invoker ↦ invokee ↦ write_m}
                          ∧ invoker ∉ Entities_compromised) ∨
                         (Flows' = Flows ∪ {invoker ↦ invokee ↦ write_m,
invokee ↦ invoker ↦ write_m} ∧ invoker ∈ Entities_compromised)
end
```

Возникновение неявных информационных потоков

Событие `pass` определяет передачу данных сущностью `z` из объекта `x` в сущность или объект `y`.

```
event pass
  any
    x
    z
    y
  where
    @grd_basic1 z ∈ Entities
    @grd_basic2 x ∈ Objects
    @grd_basic3 y ∈ Entities ∪ Objects
```

```
    @grd_basic4 x ≠ y
    @grd_basic5 (z ↦ y ↦ write_m) ∈ Flows
    @grd_basic6 (z ↦ x ↦ read_a) ∈ Accesses
    @grd_mic1 ¬((¬(integrity_level(z) ↦ integrity_level(x) ∈ le_il) ∧
                   integrity_level_r(z) ↦ integrity_level(x) ∈ le_il) ∨
   (¬(integrity_level(z) ↦ integrity_level(Resource_driver(x)) ∈ le_il) ∧
    integrity_level_r(z) ↦ integrity_level(Resource_driver(x)) ∈ le_il))
    ∨ z ∈ Entities_compromised
  then
    @act_basic1 Flows ≔ Flows ∪ {x ↦ y ↦ write_m}
end
```

Аналогично определены еще два события: `post`, описывающее неявную передачу данных сущностью `x` сущности `y` через объект `z`, и `find`, описывающее появление данных от сущности `x` в объекте `y` (или их прием сущностью `y`) посредством сущности `z`.

Распространение зоны захвата контроля

События `control_e` и `control_o` описывают захват контроля над сущностями и объектами, соответственно. В случае, если захват контроля осуществляется над драйвером объектов, контроль также захватывается и над всеми объектами, которыми управляет этот драйвер. Множество таких объектов представлено с использованием отношения, обратного функции `Resource_driver`.

```
event control_e
  any
    x
    y
  where
    @grd_basic1 x ∈ Entities \ Entities_compromised
    @grd_basic2 y ∈ Entities_compromised ∪ Objects_compromised
    @grd_basic3 y ↦ x ↦ write_m ∈ Flows
  then
    @act1 Entities_compromised ≔ Entities_compromised ∪ {x}
  @act2 Objects_compromised ≔ Objects_compromised ∪ Resource_driver~[{x}]
end
event control_o
  any
    x
  where
    @grd_basic1 x ∈ Objects \ Objects_compromised
    @grd_basic2 ∃y · y ∈Entities_compromised ∪Objects_compromised ∧
                y ↦x ↦write_m ∈Flows
  then
    @act1 Objects_compromised ≔ Objects_compromised ∪ {x}
end
```

4.3 Свойства безопасности, обеспечиваемые моделью

Основное свойство безопасности, гарантируемое моделью, утверждает, что либо возникающие информационные потоки направлены от не менее целостных сущностей или объектов, либо расширение зоны захвата контроля носит строго ограниченный характер. Ограничение заключается в том, что в этом случае в множестве захваченных компонентов

всегда уже имеется сущность с уровнем целостности большим либо равным уровню целостности компонента, к которому реализован информационный поток:

```
@main_safety_prop
    ∀u, v · u ∈ Entities ∪ Objects ∧ v ∈ Entities ∪ Objects ∧
            v ↦ u ↦ write_m ∈ Flows ⇒
      integrity_level(u) ↦ integrity_level(v) ∈ le_il ∨
        (∃w · w ∈ Entities_compromised ∧
              integrity_level(u) ↦ integrity_level(w) ∈ le_il)
```

Помимо основного свойства безопасности и всех инвариантов, представленных ранее, был сформулирован и доказан также ряд других свойств модели. Ниже приведены некоторые примеры.

Уровень целостности объекта всегда меньше либо равен уровню целостности драйвера этого объекта:

```
@driver_and_object_levels
    ∀x · x ∈ Objects ⇒
      integrity_level(x) ↦ integrity_level(Resource_driver(x)) ∈ le_il
```

Минимальный уровень целостности объектов, к которым сущность может обращаться на чтение, всегда меньше либо равен обычному уровню целостности этой сущности:

```
@integrity_levels_prop
    ∀x · x ∈ Entities ⇒ integrity_level_r(x) ↦ integrity_level(x) ∈ le_il
```

Также был определен ряд свойств касаемо иерархии объектов и так далее. Возможность автоматизированно доказывать эти свойства позволила выполнить проверку выполнимости любого интересующего свойства.

На основе сформулированных инвариантов было сгенерировано около 300 теорем (proof obligations). При этом доказательство 54% из них потребовало ручного вмешательства. Однако в большинстве случаев достаточным оказалось проведение несложных манипуляций и вызов программных решателей и SMT-солверов. Однако иногда процесс доказательства был трудоемким, состоящим из множества шагов (упрощений, применений правил логического вывода, разбиений на случаи, введения дополнительных лемм). Особенно можно отметить случаи, когда доказываемая теорема неверна. В этих случаях процесс ее доказательства позволял прийти к утверждениям, достаточно ясно проясняющим проблему, что, в частности, привело к более глубокому пониманию модели. Был исправлен ряд неточностей начальных версий модели на естественном и математическом языках.

Таким образом, возможность автоматизированной проверки выполнения свойств модели позволила доказать множество теорем и провести большое количество экспериментов по модификации модели и определению ее свойств, что при ручном подходе было бы чрезмерно трудоемким. С другой стороны, доказательство в Rodin представляется в виде большого дерева, отражающего, в том числе, множество механизированных шагов доказательства. Видится, что такое представление не может быть полноценной заменой ручного доказательства в том смысле, что последнее может рассматриваться как ясное и лаконичное объяснение, почему выполняются основные свойства модели.

5. Заключение

В работе формализована модель мандатного контроля целостности, реализованная в микроядерной операционной системе KasperskyOS. Выбранный язык Event-B позволил получить формальное описание модели, достаточно близкое к исходному описанию на математическом и естественном языках, и при этом свободное от ошибок и неточностей, которые возникали при использовании естественного языка. Сформулированы и доказаны свойства безопасности, обеспечиваемые моделью. Опыт доказательства этих свойств

является положительным. Достаточно высокая степень автоматизации при доказательстве позволила найти ответы на множество вопросов касаемо свойств модели, что было бы чрезмерно трудоемко при ручном подходе. В то же время, для детального объяснения, почему выполняются ключевые свойства модели, ручное доказательство видится более подходящим.

Список литературы / References

[1]. Jaeger T. Operating System Security. Morgan and Claypool Publishers, 2008, 220 p.

[2]. Landwehr C. Formal Models for Computer Security. ACM Computing Surveys, vol. 13, issue 3, 1981, pp. 247-278.

[3]. Федеральная служба по техническому и экспортному контролю. Информационное сообщение о требованиях по безопасности информации, устанавливающих уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий от 29 марта 2019 г. / Federal Service for Technical and Export Control. Announcement on Information Security Requirements Establishing Levels of Confidence in Information Protection Means and Information Technology Security Means. URL: https://fstec.ru/normotvorcheskaya/informatsionnye-i-analiticheskie-materialy/1812-informatsionnoe-soobshchenie-fstek-rossii-ot-29-marta-2019-g-n-240-24-1525, accessed 2020-04-16 (in Russian).

[4]. Буренков В.С., Кулагин Д.А. Модель мандатного контроля целостности в операционной системе KasperskyOS. Труды ИСПРАН, том 32, вып. 1, 2020 г., стр. 27-56 / Burenkov V.S., Kulagin D.A. A Mandatory Integrity Control Model for the KasperskyOS Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020, pp. 27-56 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–2.

[5]. Abrial, J.-R. Modeling in Event-B: System and Software Engineering. Cambridge University Press. 2010, 612 p.

[6]. Event-B and the Rodin Platform. URL: http://www.event-b.org/, accessed 2020-04-16)

[7]. Kozachok A.V. TLA+ based access control model specification. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 5, 2018, pp. 147-162. DOI: 10.15514/ISPRAS-2018-30(5)-9.

[8]. Kim I., Kang M., Choi J., Zegzhda P. Formal Verification of Security Model Using SPR Tool. Computing and Informatics, vol. 25, no. 5, 2006, pp. 353-368.

[9]. Hu V., Kuhn D., Xie T., Hwang J. Model Checking for Verificaton of Mandatory Access Control Models and Properties. International Journal of Software Engineering and Knowledge Engineering, vol. 21, no. 1, 2011, pp. 103-127.

[10]. Zhang N., Ryan M., Guelev D. Evaluating Access Control Policies Through Model Checking. In Proc. of the International Conference on Information Security, 2005, pp. 446-460.

[11]. Devyanin P., Khoroshilov A., Kuliamin V., Petrenko A., Shchepetkov I. Using Refinement in Formal Development of OS Security Model. Lecture Notes in Computer Science, vol. 9609, 2015, pp. 107-115.

[12]. Devyanin P., Khoroshilov A., Kuliamin V., Petrenko A., Shchepetkov I. Formal Verification of OS Security Model with Alloy and Event-B. Lecture Notes in Computer Science, vol. 8477, 2014, pp. 209-313.

[13]. Девянин П.Н., Ефремов Д.В., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В.. Моделирование и верификация политик безопасности управления доступом в операционных системах. М., Горячая линия – Телеком, 2019, 214 стр. / Devyanin P.N., Efremov D.V., Kulyamin V.V., Petrenko A.K., Khoroshilov A.V., Shchepetkov I.V. Modeling and verification of access control security policies in operating systems. M., Hotline – Telecom, 2019, 214 p. (in Russian),

[14]. Девянин П.Н., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. Интеграция мандатного и ролевого управления доступом и мандатного контроля целостности в верифицированной иерархической модели безопасности операционной системы. Труды Института системного программирования РАН, том 32, вып. 1, 2020, стр. 7-26 / Devyanin P.N., Kuliamin V.V., Petrenko A.K., Khoroshilov A.V., Shchepetkov I.V. Integrating RBAC, MIC, and MLS in Verified Hierarchical Security Model for Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 7-26 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–1.

[15]. Hussain S., Farid S., Alam M., Iqbal S., Ahmad S. Modeling of Access Control System in Event-B. The Nucleus, vol. 55, no. 2, 2018, pp. 74-84.

[16]. Romanovsky A., Thomas M. (Editors). Industrial Deployment of System Engineering Methods. Springer-Verlag, 2013. 274 pp.

[17]. Alves-Foss J., Oman P., Taylor C. The MILS Architecture for High-Assurance Embedded Systems. International Journal of Embedded Systems, vol. 2, no. 3/4, 2006, pp. 239-247.

Информация об авторе / Information about the author

Владимир Сергеевич БУРЕНКОВ – кандидат технических наук, разработчик-исследователь. Сфера научных интересов: модели программно-аппаратных систем, формальные методы верификации.

Vladimir Sergeevich BURENKOV – PhD, research developer. Research interests: models of computer systems, formal verification methods.

DOI: 10.15514/ISPRAS-2020-32(6)-4

A Formal Model of a Partitioned Real-Time Operating System in Promela

S.M. Staroletov, ORCID: 0000-0001-5183-9736 <serg_soft@mail.ru>
Polzunov Altai State Technical University,
Lenin avenue 46, Barnaul, 656038, Russia

Abstract. Real-time partitioned operating systems meet the current avionics standard of reliable software; they are capable of responding to events from devices with an expected speed, as well as sharing processor time and memory between isolated partitions. Model-based Checking is a formal verification technique in which a software model is developed and then it is automatically checked for the compliance with formal requirements. This method allows proving the correct operation of the model on all possible input data, all possible ways of processes switching and interactions. In this article, we describe a formalized model of an open-source partitioned operating system POK. We implement the model in Promela language for SPIN tool with the purposes of formal verification using the Model Checking method. The model is designed to describe the behavior of: partition and process schedulers, system calls through a software interrupt, kernel libraries for working with synchronization primitives and processes awaiting, user code which consists of several processes in different partitions that are synchronized through a semaphore. The described approach can be used to verify the correct synchronization, the proper operation of the scheduler algorithms, and the accurate data access from different partitions by introducing the corresponding requirements in the form of formulas of the linear-time temporal logic.

Keywords: formal verification; operating system; partitioned system; real-time system; model checking; system programming; Promela; SPIN

For citation: Staroletov S., A Formal Model of a Partitioned Real-Time Operating System in Promela. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 49-66. DOI: 10.15514/ISPRAS-2020-32(6)-4

Формальная модель партицированной операционной системы реального времени на Promela

С.М. Старолетов, ORCID: 0000-0001-5183-9736 <serg_soft@mail.ru>
Алтайский государственный технический университет им. И.И. Ползунова,
656038 Барнаул, проспект Ленина, 46

Аннотация. Текущий стандарт надежного программного обеспечения для бортовых контроллеров – это многораздельная операционная система реального времени, которая способна реагировать на события от устройств с ожидаемой скоростью, а также делить процессорное время и память между изолированными разделами. Верификация на основе модели – это метод формальной проверки программного обеспечения, при котором разрабатывается программная модель, а затем она автоматически проверяется на соответствие формальным требованиям. Этот метод позволяет доказать правильность работы модели на всех возможных входных данных, всех возможных способов переключения процессов и взаимодействий. В этой статье описывается формализованная модель открытой многораздельной операционной системы POK, реализованная на языке Promela средства SPIN для формальной верификации методом Model Checking и предназначенная для моделирования поведения: планировщика разделов и процессов; системных вызовов через программное прерывание; библиотеки ядра для работы с примитивами синхронизации и ожиданием процессов; пользовательский код, осуществляющий работу нескольких процессов в разных разделах, которые синхронизируются

через семафоры. Данный подход может быть использован для проверки корректности синхронизации, работы алгоритмов планировщика, корректного доступа к данным из разных разделов путем задания соответствующих требований в виде формул темпоральной логики линейного времени.

Ключевые слова: формальная верификация; операционные системы; партицирование; ОС реального времени; Model Checking; системное программирование; Promela; SPIN

Для цитирования: Старолетов С.М. Формальная модель партицированной операционной системы реального времени на Promela. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 49-66 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(6)–4

1. Introduction

This work is a part of a project to provide verification methods for controllers of cyber-physical systems with high-reliability requirements [1]. In this paper, we refer to a concept of partitioned operating systems, mostly related to avionics software standards. The main goal is to develop and verify software based on existing open-source solutions, as well as to apply the results as a model for teaching the courses *«Components of operating systems»* and *«Software verification»*.

In this paper, we follow the creation of a model for *POK* (Partitioned Operating System Kernel) [2]. Using its source code, we create a corresponding code in *Promela* [3], an input language of *SPIN* verifier. On the one side, the language offers to encode real algorithms close to original C implementation, but on the other side, this language has a clear formal semantic and the model in this language can (without any shortcomings) be translated to a Kripke structure and then verified by querying LTL formulas with temporal properties of desired OS model behavior.

This publication has the following structure: in Section 2, we briefly describe the POK concept and model checking with SPIN; in Section 3, we show the core of presented approach, how to model a client program using an emulation of the instruction pointer; in Section 4, we highlight our scheduling model; in Section 5, we present ways to model the syscalls; in Section 6, we browse some existing solutions in this area; in Section 7, we discuss the solution and finally, in Section 8, we make a conclusion and give a link to our resulting open-source model in Promela.

The main contributions of the paper are: (a) we show the applicability of Promela to model OS behavior; (b) we create an executable model of a partitioned OS.

2. Background

2.1. A Concept of Partitioned Real-time OS

A BSD-licensed open-source OS POK, which satisfies avionics software standards with some limitations, was created at a research institute in France as a PhD thesis by Julien Delange [4], it applies the Model-Driven Engineering approach [5] for describing the system configurations, and its source code is available in [2].

We have already summarized in [1] its main features as:

- *MDE approach*: initial OS kernel configuration in AADL language [6] with code generation and a possibility to represent the configuration graphically;
- it is a good *proof-of-concept* with a set of working models and examples;
- partially conforms to the *ARINC 653* real-time onboard aviation system standard [7];
- *protected partitions* with time and memory space resources isolation;
- real-time processes schedulers with different *strategies of two types*: (a) partition planner (b) process planner in each partition;
- *controllable* port and message interactions between processes; also the BlackBoard concept [7] is used.

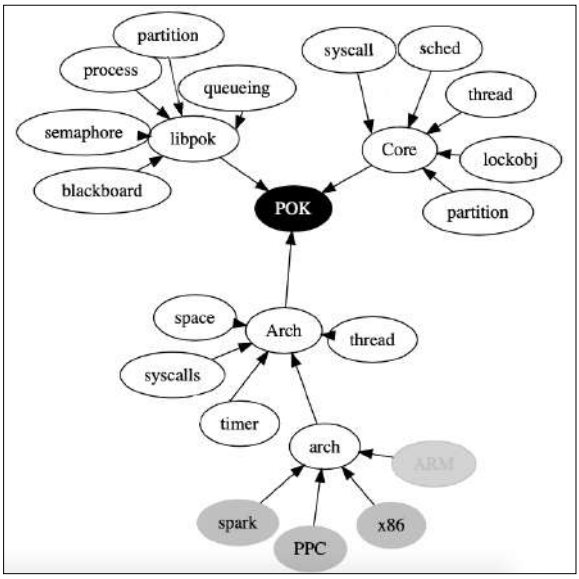

Fig. 1. A scheme of POK internal architecture

The use of OS, which is designed according to avionics standards and provides the isolation and verifiable interprocess communication, increases the robustness of the functionality of a cyber-physical system at the system level.

By browsing the source code [2], we created a scheme of internal POK architecture, shown in fig. 1. It comprises three principal layers:

- *Arch* with platform-dependent code (open-source repository includes realization for three platforms: x86-qemu, PowerPC and Sparc), also there are some works on an ARM port;
- *Core* for internal kernel code, syscalls processing;
- *libpok* can be used to call from the user's code as an API.

The ARINC 653-compatible API offers to work with partitions, processes, locking objects, ports, queries and messages in a standardized and certifiable way. The API is a high-level abstraction, in this paper, we do not touch it, and we proceed to model low-level things on which it is all based.

2.2. Model Checking with the SPIN tool

SPIN [8, 9] is a utility for verifying the correctness of distributed software models. The abbreviation SPIN stands for Simple Promela INterpreter. The SPIN system checks not the programs themselves, but their models. To build a model for an original parallel program or an algorithm, the verifying engineer (usually manually) creates a representation of this program in the C-like input language, called Promela (PROtocol MEta-LAnguage) [10].

To deal with the problem we are formalizing, we may rely upon the following language features [11]:

- it is an actor-based (process- and message-oriented) language;
- it is primarily designed to describe protocols and interoperations;
- it has C-styled syntax and fix-size finite data types;
- it uses function inlining quite similar to the macros in C;
- it allows custom types definition (using typedef as similar in C);
- it introduces "atomic" sections to model code that is running in parallel without any context switching inside.

There are no pointers, so special techniques should be used here to provide abstractions for them. For a long time, the language has been used mainly in academia, but instantly, the language authors added syntax constructions to describe complicated programs, the project has moved to GitHub and modern modular text editors (like Visual Studio Code) introduced support to highlight, refactor and run programs in Promela.

As a result, we think that it is a suitable language to model OS internals with the aim of further formal verification. Promela constructs are simple, they have clear and distinct semantics, which allows the verifier to translate any program in this language into a verifiable transition system with a finite number of states. The requirements for the model are expressed in LTL (Linear-time Temporal Logic) [12].

The model checking process inside comprises (a) converting a model program into a Büchi automaton by considering the change in its state, (b) resolving non-determinism, (c) modeling context-switching as the creation of variants of possible transitions, (d) converting the negation of a temporal formula of a requirement into an automaton, and (e) creating the resulting parallel composition of automata [11]. During the verification process, a traversing is made through all the states of the resulting automaton, plus at the same time, violations of requirements are checked as generated asserts. If the requirement is violated, the verifier produces a counterexample as a sequence of control states of the system (a trail) which points to the violation of the requirement.

3. OS Internals Modeling: Our Approach

3.1. The Sample to Study

To model the partitioned OS, we carefully studied an example with multi-threaded work of processes (located at examples/semaphores in [2], see fig. 2). We setup a C development environment with prescribed source and include paths. We walked through the source code and inspected all called functions or macros. This made it possible to recreate the behavior of a real OS.

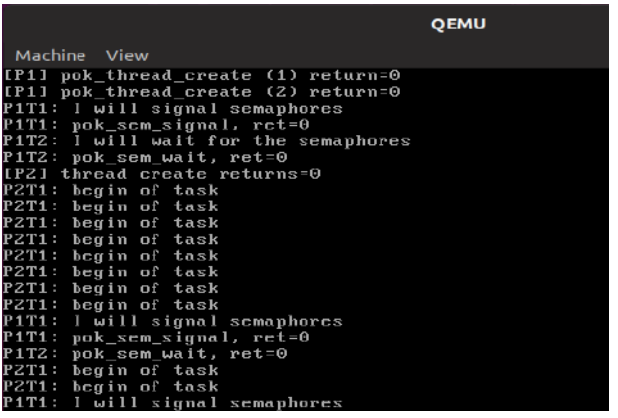

Fig. 2. Minimal partitioned code example from [2], working in QEMU environment.

In Listing 1, we show part of the source activity code of the Thread 1, working in the Partition 1.

```
void* pinger_job () {
   pok_ret_t ret;
   while (1) {
      printf ("P1T1: I will signal semaphores\n");
      ret = pok_sem_signal (sid);
      printf ("P1T1: pok_sem_signal, ret=%d\n", ret);
      pok_thread_sleep (2000000);
   }
}
```

Listing 1. A multi-threaded sample

In the code, one thread signals a semaphore and sleeps, and continues to do it forever. The other threads at the same time wait for the semaphore and do some work.

We choose this sample because of two things:

- it is really a minimal behavior of a partitioned OS;
- it contains multi threads, multi partitions as well as locking primitives and sleeping, so it is suitable to model dynamic scheduling algorithms.

3.2. Modeling the Activity Code in Promela

In Listing 2, we present a model for the above code.

```
proctype threadP1T1(short myPartId; short myThreadId) {
do
::(osLive == 1) ->
atomic {
 if ::(currentPartition == myPartId
  && currentThread == myThreadId && currentContext.IP == 0) ->
     {
      pok_print(P1T1_I_will_signal_semaphores);
      currentContext.IP++;
     }
   ::else ->
  if ::(currentPartition == myPartId &&
 currentThread == myThreadId && currentContext.IP == 1) ->
     {
      pok_sem_signal(sid, currentContext.r0);
      currentContext.IP++;
     }
  ::else ->
 if ::(currentPartition == myPartId &&
  currentThread == myThreadId && currentContext.IP == 2) ->
    {
     pok_printf(P1T1_pok_sem_signal_ret, currentContext.r0);
     currentContext.IP++;
    }
  ::else ->
 if ::(currentPartition == myPartId &&
  currentThread == myThreadId && currentContext.IP == 3) ->
    {
     pok_delay(2000);
     currentContext.IP = 0; /* inf loop */
    }
  ::else -> skip;
   fi
  fi
 fi
fi
}
::else -> break;
od
}
```

Listing 2. Model for the multi-threaded sample

Here we see a state machine that makes transitions between its states. A state of the process is characterized by the *IP* (instruction pointer) register. There are also the guard conditions to check if we are the current one to execute. The main idea here: *all the processes are traversing thought their states if they are active, and the OS scheduler is activated periodically and selects a current partition*

as well as a current process (changes the currentPartition and currentThread variables), and that causes the whole system model to run.

Also, there are *pok_print*, *pok_delay*, *pok_sem_signal* macros that emulate the syscalls in the OS, we consider them in the appropriate section.

As a result of the current section, we can state that any code that models some actions in a real OS must satisfy the following properties:

- for each thread in the system, a corresponding *process* is created in Promela;
- for all his calculations, it uses *only register variables* from the current context;
- after each line of significant code, the register *IP is incremented*;
- each line of code is executed *in the switch* by IP, current process and current partition.

3.3. Data Definition in the Model

Thanks to the support of *typedef* complex structures and arrays in Promela, we can build mostly a normal data definition in our model (see fig. 3). Here we introduce *Context*, *Thread*, *Partition* and *Semaphore* structures to model corresponding OS entities.

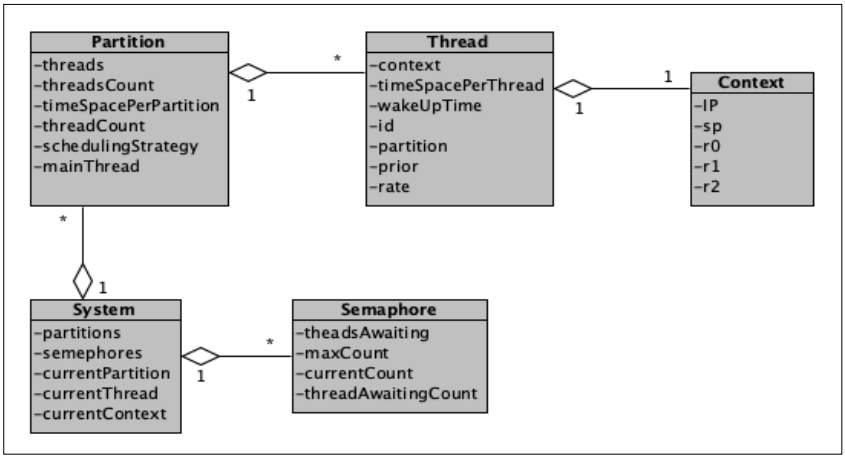

Fig. 3. Data structures in our model

To simulate the code execution, we explicitly introduce the processor registers as Promela variables and put them into the current execution context, which simulates one processor with its memory. Those are primarily a register for the current instruction pointer (*IP*), a stack register (*sp*) and several arithmetic registers (r_0-r_n), see Listing 3.

```
typedef Context {
    int IP;          //instruction pointer
    int sp;          //stack pointer - for further modeling
    int r0;          //arithmetic registers
    int r1;
    int r2;
}
```

Listing 3. Model for the state of the current thread

IP is used for the program flow in a thread (see Listing 2), arithmetic registers should be used in calculations, *sp* is added for future use (for example, to model local memory, procedures and parameter passing). Then we include such a context to the thread definition (see Listing 4).

```
typedef Thread {
    Context context;  //thread context to save
    short timeSpacePerThread; //count of ticks to run
    bit isLocked;     //1 if it has been locked on a semaphore
```

```
    int wakeUpTime;    //wake up time to schedule using 'sleep'
    short id;          //unique thread id
    short partition;   //number of the parent partition
    short prior;       //for further model with priorities
    short rate;        //current execution time - for rms
}
```

Listing 4. Data definition for threads

A thread is characterized by its context, some parameters and time space (amount of time to run the thread before the switch). Now and after we are going to count *time* in ticks, countable by the scheduler. After all, we introduce the partition definition as shown in Listing 5.

```
typedef Partition {
    short timeSpacePerPartition; //count of ticks to run
    short threadCount;
    Thread threads[MAXTHREADS];  //threads of this partition
    short schedulingStrategy;    //type of sched for threads
    short mainThread;            //first thread to run
}
```

Listing 5. Data definition for partitions

It consists of a number of threads, time space for the partition to run between a switch, scheduling strategy of related threads and the main thread to peak at first.

4. Modeling the Scheduler

For the first iteration, we show a simple non-deterministic scheduler that randomly selects a partition of two and a thread of two inside, see Listing 6.

```
proctype schedNonDeterministicInstance() {
 do
 :: realTime < MAXTIMESIM -> {
  atomic {
    saveCurrentContext();

    //non-deterministic partitions scheduler
    if
      ::true -> currentPartition = 0;
      ::true -> currentPartition = 1;
    fi

    if
      ::(currentThread == 0) -> currentThread = 1; //stub
      :: else -> currentThread = 0;
    fi
      realTime++;
      restoreCurrentContext();
   }
  }
 :: else -> {
      printf("Simulation time is over!\n");
      osLive = 0;
      break;
    }
 od
}
```

Listing 6. Simple scheduler that peaks random partitions and threads

The scheduler runs as a Promela process; it activates at some random time. The system runtime is bounded to a constant, and the *realTime* variable is used to count time passed in the whole system, so we are counting time right once the scheduler is activated (corresponds to the hardware timer

interrupt handling). The s*aveCurrentContext* and r*estoreCurrentContext* macros are used to save the current context to a place of the context of a current thread and restore it respectively. So, using ideas in Listing 2 and Listing 6, one can implement a very simple model of the scheduling.

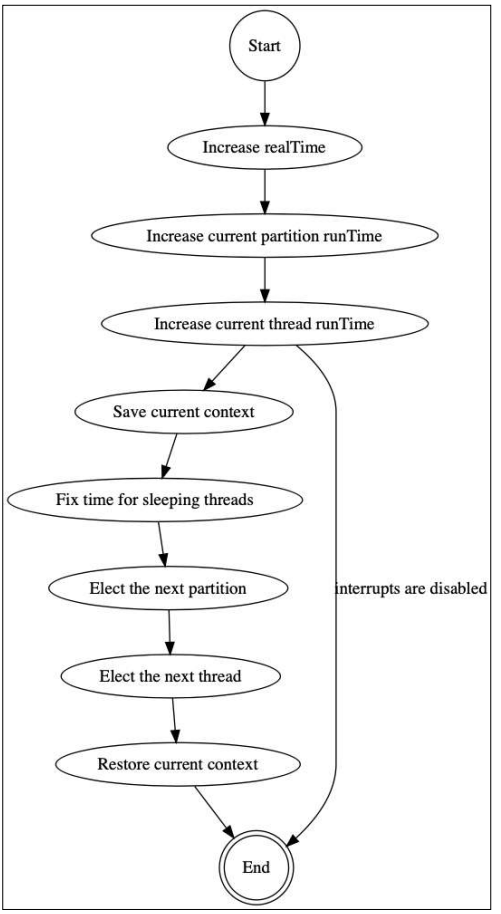

Fig. 4. A model of a partitioned scheduler

The real scheduler that we use in our model is much more sophisticated. In fig. 4 we depicted its block scheme. It runs in a loop that is fired on don-deterministic times. The first thing to do is to increment the time variables of the whole system as well as time running of a current partition and a current thread (remember, we have bounds for these times in the thread and partition definition structures). Then only if a logical variable for disabled interrupts is not set (corresponds to disabling the interrupts in the real OS), we continue to the switching process. The next thing to do — is to fix wakeup time for all the sleeping threads. That means that for all threads with elapsed time of sleeping we should remove their sleeping statuses (because we had already changed current time and some threads have just become candidates to switch to). The resting behavior of the scheduler is the same as the previous one: save current context, elect a partition, elect a thread and restore the context. However, here we do the elections according to set election strategies and current locking statuses.

In Listing 7, we show a piece of code to elect the next thread. Here we introduce scheduling strategies that are set in the partitions during the initialization phase. Then the right strategy can be applied in the scheduling loop.

```
mtype = {sched_part_rms_strategy, sched_part_rr_strategy,
sched_part_edf_strategy, sched_part_llf_strategy}
```

```
inline elect_next_thread(needPeakAThread) {
if
::(partitions[currentPartition].schedulingStrategy ==
 sched_part_rms_strategy) -> sched_part_rms(needPeakAThread);
::(partitions[currentPartition].schedulingStrategy ==
 sched_part_rr_strategy)  -> sched_part_rr(needPeakAThread);
::(partitions[currentPartition].schedulingStrategy ==
 sched_part_llf_strategy) -> sched_part_llf(needPeakAThread);
::(partitions[currentPartition].schedulingStrategy ==
 sched_part_edf_strategy) -> sched_part_edf(needPeakAThread);
::else -> skip;
fi
}
```

Listing 7. An extensible thread election.

In Fig. 5 we show our implementation of the round-robin thread election.

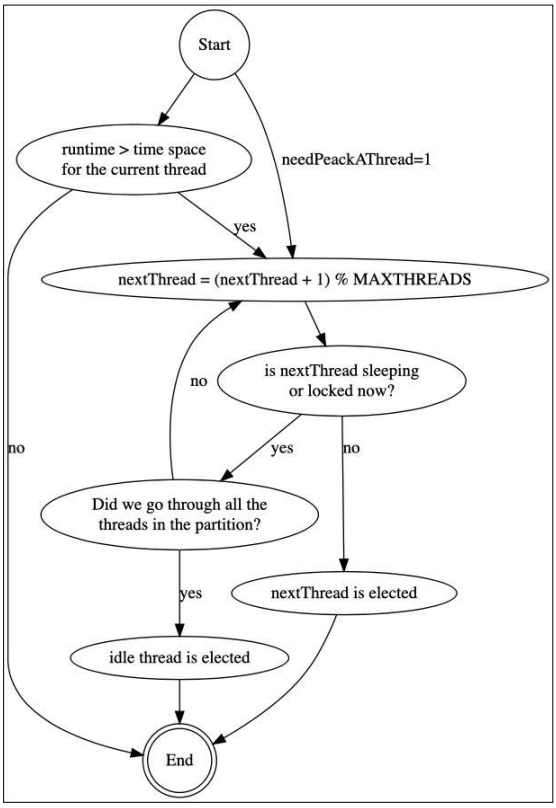

Fig. 5. A model of a round-robin scheduler with the possibility of locking and sleeping

In a loop, we select a next thread and check whether it is runnable (it means it can sleep after requesting to sleep in the code or to be locked on a semaphore). If we are not able to select the next thread, we select the virtual one (idle thread) with no code to execute. The guard conditions to start the elections are (a) the current thread has run out of its time or (b) the scheduler is asked to select a new thread (due to sleep or locking have queued).

To select a new partition, we only use the condition that the current partition has run out of its time.

5. Modeling the SysCalls

In POK, all API that OS provides to its client processes (for example, creating a semaphore, waiting for it or blocking it) are done through system calls. This means that for each interaction with a kernel object, the generation of a software interrupt is performed. This approach allows to control such calls from the OS, to be able to prioritize them, to perform them in a protected context.

During the modeling, we create an enumeration of possible syscalls, available to the user. Then we create macros to wrap API calls in a syscall executor routine (it fully compliments to the POK code). The executor prepares the syscall parameters in registers and generates a software interrupt. We model it as a message passing to a Promela channel. The syscall executor puts a signal to the channel and the syscall handler waits here for the signal in a loop, awakens and actually performs the call. In Listing 8 we demonstrate the user syscall library (see also Listing 1 for the example of its utilization).

```
//syscalls types
mtype = {syscall_sem_p, syscall_sem_v, syscall_delay,
syscall_printf}

//library available to user
inline pok_sem_signal(sid, ret) {
    printf("pok_sem_signal\n");
    pok_do_syscall(syscall_sem_v, sid, NOPARAM, ret);
}

inline pok_sem_wait(sid, ret) {
    printf("pok_sem_wait\n");
    pok_do_syscall(syscall_sem_p, sid, NOPARAM, ret);
}
```

Listing 8. Syscalls user library

In Listing 9 we show the model of syscalls executor.

```
inline pok_do_syscall(N, param1, param2, ret) {
  atomic {
    //pass the params
    currentContext.r0 = N;
    currentContext.r1 = param1;
    if
        ::(param2 != NOPARAM) -> currentContext.r2 = param2;
        ::else -> skip
    fi
    }
    //emit the interrupt
    InterruptController ! POK_INTERRUPT;
    //wait for iret
    InterruptRet ? ret;
}
```

Listing 9. Syscalls executor

In Listing 10, we show part of the model of syscalls handler with a switch by a syscall id.

```
interruptsDisabled = 1; //stop the scheduler (soft model)
saveCurrentContext();
if
 ::(intNum == POK_INTERRUPT) -> {
   if
     ::(id == syscall_sem_v)  -> sem_signal(param1);
     ::(id == syscall_sem_p)  -> sem_wait(param1);
     ::(id == syscall_delay)  -> sleep(param1);
     ::(id == syscall_printf) -> print(param1, param2);
     ::else -> skip; //unknown syscall id
```

```
    fi
  }
  ::else -> skip;
fi
restoreCurrentContext();
```

Listing 10. A part of syscalls handling

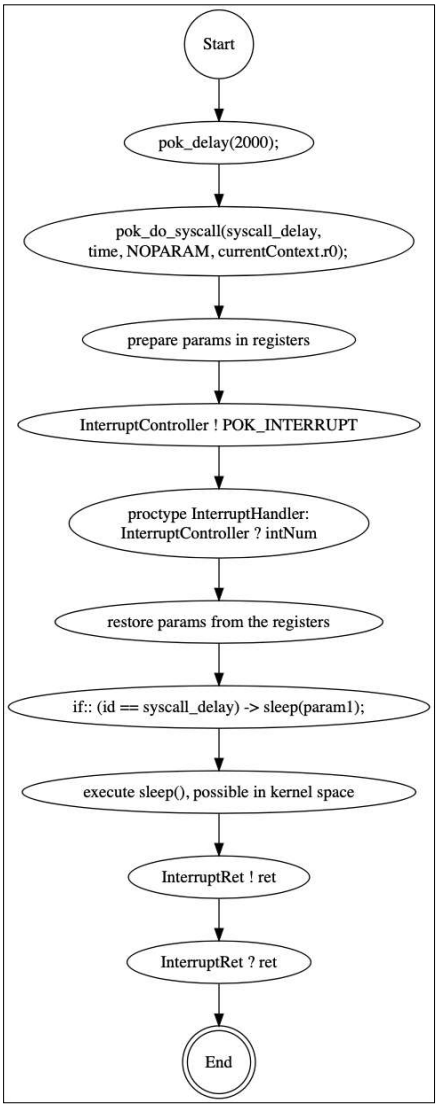

Fig. 6. Syscalls processing model

The handler gets the parameters from the registers and uses the switch operator to decide which kernel function it should execute. The overall scheme of interrupts modeling is presented in fig. 6.

In addition, we focus on the implementation of our syscalls:

- *sleep* is implemented using the calculation of a wakeup time for the current thread based on the given delay value and call the scheduler;
- *wait on a semaphore* is implemented by updating the semaphore counter and adding the current thread to a list of awaiters of the given semaphore if it is necessary;

- *signal a semaphore* is implemented by updating the semaphore counter and removing the current thread from a list of awaiters of the given semaphore if it is necessary;
- *print* is implemented using switching by the parameter and printing a corresponding string.

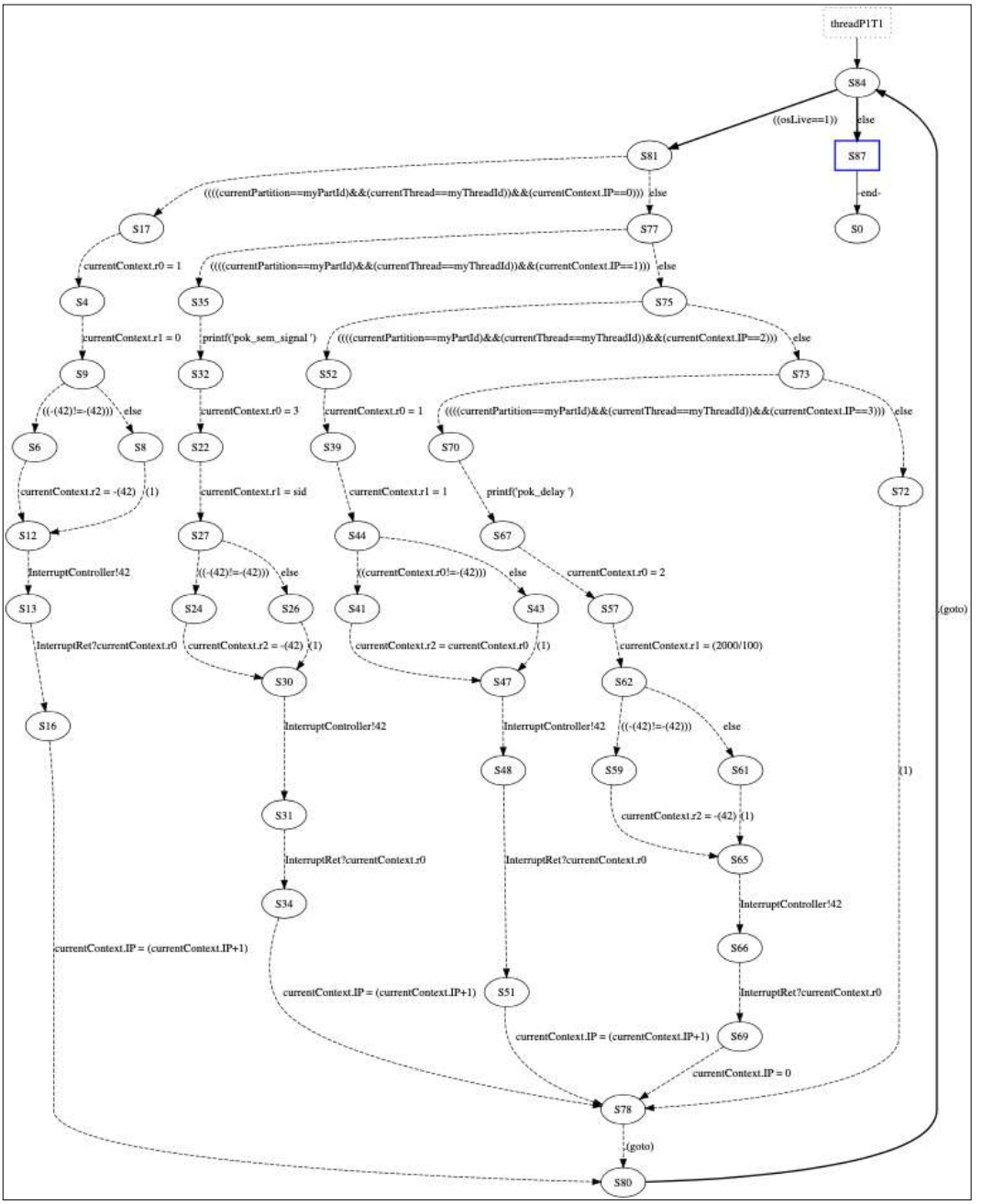

Fig. 7. Automaton representation of a process in our OS model.

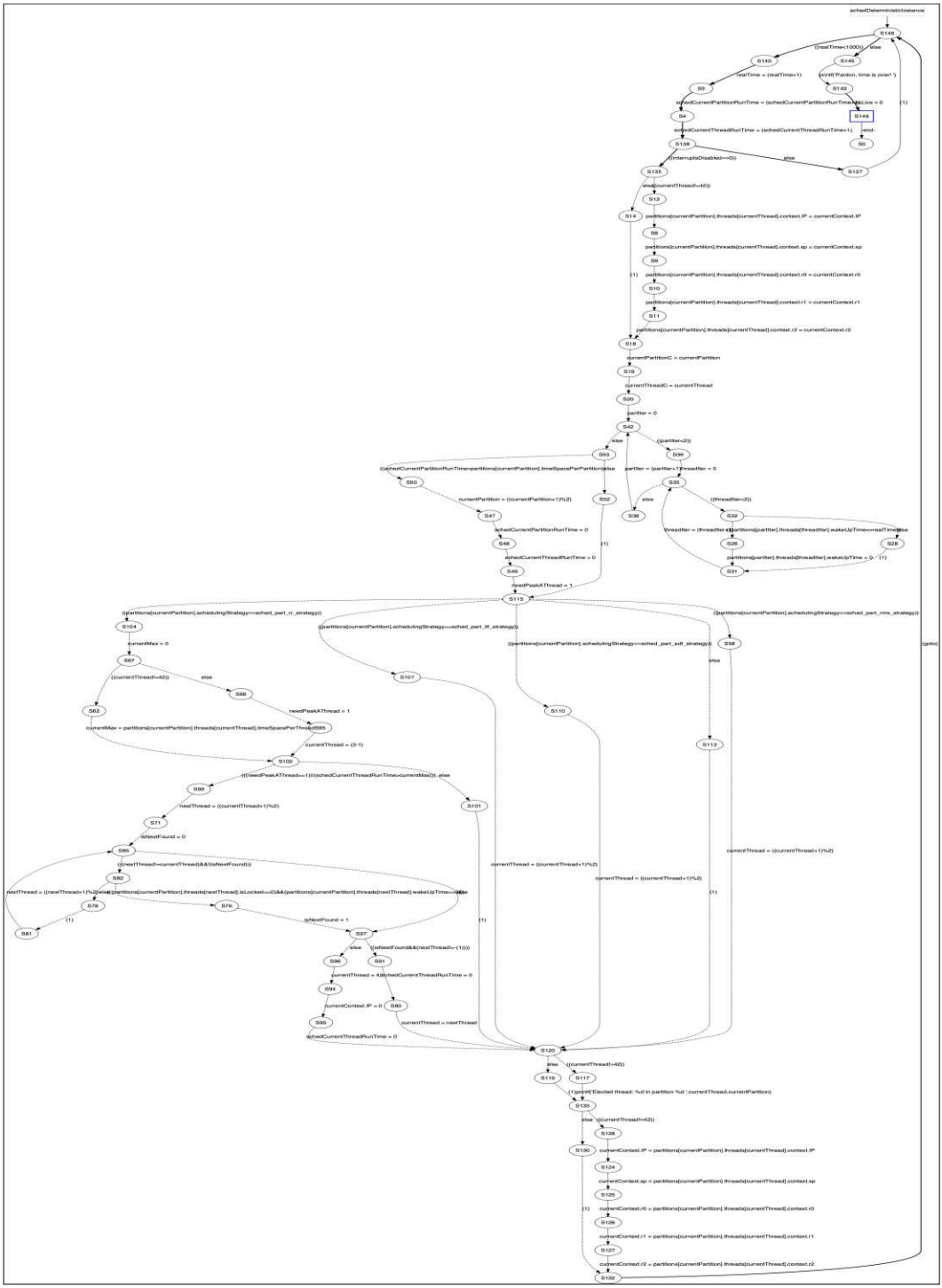

Fig. 8. Automaton representation of the scheduler in our OS model.

6. Related work

Today's OS for reliable cyber-physical systems like flight machines should be definitely real-time and must offer memory space and time division capabilities. There exists the avionics industry-related standard for these requirements, created by the Aeronautical Radio, Inc., ARINC 653 [7].

Methods of validation of ARINC architectures were given in [13]. Some techniques on verification of OS (mostly related to Linux) were presented by the ISP RAS in [14-16].

In Russia, a certified partitioned operating system intended for the aircraft, was created by GosNIIAS and ISP RAS with advanced debugging capabilities, rewritten scheduler, system partition feature and different platforms support [17, 18], some results were GPLv3 licensed.

The most famous approach to verification of OS is presented in [19]. The authors created executable specifications of an L4 microkernel in Haskell based on initial C implementation and then refined them into an Isabelle/HOL model. There is also a good literature review on this area in their paper. Our approach follows theirs: we created an executable specification in Promela according to the C code of POK, but then we are going to use model checking methods instead of theorem proving.

7. Discussions

7.1. Visualization of the Model

Using the SPIN capability to export model automata as .dot diagrams (*./pan -D* [11]), we created the automata representations of a process (see fig.7) as well as of the scheduler (see fig.8). These images are presented here to estimate the complexity of resulting automata.

The process automation (corresponds to Listing 1) consists of about 80 states, we can see that some states are duplicated due to inline macros to execute the syscalls (see Section 5). The scheduler automaton consists of about 150 states.

Building such automata by hand is very costly, so the executable specification in Promela really helps to obtain a formal model to provide further checks. In addition, the model is very extensible, it is easy to add scheduling strategies (see Listing 7 for the reference) as well as implement additional syscall types, etc.

7.2. Simulation of the Model

In fig. 9 we depict a simulation process of our model, using the command-line SPIN run.

```
        Elected thread: 0 in partition 0
[1] P1T1: I will signal semaphores
            pok_sem_signal
        Elected thread: 1 in partition 0
        Elected thread: 0 in partition 0
[3] P1T1: pok_sem_signal_ret = 1
        Elected thread: 1 in partition 0
[5] P1T2: I will wait for the semaphores
            pok_sem_wait
[5] P1T2: pok_sem_wait_ret = 1
        Elected thread: 0 in partition 0
        Elected thread: 1 in partition 0
            pok_sem_wait
[6] P1T2: pok_sem_wait_ret = 1
            pok_delay
Elected thread: 0 in partition 0
[7] P1T1: I will signal semaphores
            pok_sem_signal
        Elected thread: 0 in partition 0
            pok_sem_signal
[9] P1T1: pok_sem_signal_ret = 1
            pok_delay
        Elected thread: 0 in partition 1
[101] P2T1: begin of task
                pok_delay
        Elected thread: 0 in partition 0
        Elected thread: 1 in partition 0
            pok_delay
Elected thread: 0 in partition 0
[2022] P1T1: I will signal semaphores
            pok_sem_signal
```

Fig. 9. Model simulation using SPIN

We see that the processes work as expected (we apply the soft real-time strategy here), the scheduler does partitions as well as threads switching, and the processes wait expected time then do the interprocess communications using the semaphore.

7.3. On the Model Verification

Although we are still thinking about properties for verification, in this subsection we provide a way to check the correctness of partitions switching. According to our model, we use some string constants (see the parameters to *pok_print* in Listing 1). Since we know which partition each string belongs to, we can set the expected match of each string to its partition, see Listing 11.

```
//string constants
#define P1T1_I_will_signal_semaphores 0
#define P1T1_pok_sem_signal_ret 1
#define P1T2_I_will_wait_for_the_semaphores 2
#define P1T2_pok_sem_wait_ret 3
#define P2T1_begin_of_task 4

//map data-partition
short partitionByDataIndex[5] = {
        PARTITION1,
        PARTITION1,
        PARTITION1,
        PARTITION1,
        PARTITION2
};
```

Listing 11. Output strings and a mapping to the partitions

Then, as we know the expected partition matching and will know the actual one when we run the model, we provide the following check macro, see Listing 12.

```
//check if we are in the correct partition
inline checkPointer(expectedPartition, actualPartition) {
    if
        :: (expectedPartition != actualPartition) ->  {
            pointersOk = 0;
            printf("segmentation fault!\n");
        }
        :: else -> skip
    fi
}
```

Listing 12. A macro to check the safety of data strings

And the macro *checkPointer* is now used in the implementation of *print* syscall, see Listing 13.

```
inline print(string, param) {
 checkPointer(partitionByDataIndex[string], currentPartition);
 if
  ::(string == P1T1_I_will_signal_semaphores) ->
    printf("[%d] P1T1: I will signal semaphores\n", realTime);
  ::(string == P1T1_pok_sem_signal_ret) ->
  printf("[%d] P1T1: pok_sem_signal_ret = %d\n", realTime, param);
   ...
  ::else -> skip
 fi
}
```

Listing 13. A fragment of print syscall implementation

Therefore, the verification process will be checking an LTL formula «*it is always that pointers are ok*»:

$$\boldsymbol{G}\ (pointersOk\ ) \tag{1}$$

where *pointersOk* variable can be changed when a string is requested from an incorrect partition (see Listing 12) during all possible runs of the model.

8. Conclusion

In this paper, we analyzed the purpose, composition, and structure of the partitioned real-time OS. We discussed a possible approach for creating a model of such an OS in Promela. The proposed solution allows us to move from complex architecturally dependent code in C to general operating system behavioral models that can be used in the education process. Also, having a formalized OS model, we can check the security properties of code execution in partitioned systems with a possibility to apply different scheduling algorithms.

Possible future steps are:

- modeling of different scheduling strategies;
- modeling of ARINC API [7];
- creation of control variables, construction of LTL formulas and model verification;
- multicore scheduling models;
- models both for the hard real-time as well as the soft real-time using scheduling strategies;
- checking of cyber-physical models running in such an OS using our library [20].

The current code for the implementation of the approach described in this paper is freely available in [21]. The authorship is registered in the database of the Federal Institute of Industrial Property [22].

References / Список литературы

[1]. Staroletov S.M., Amosov M.S., Shulga K.M. Designing robust quadcopter software based on a real-time partitioned operating system and formal verification techniques. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 4, 2019. pp. 39-60. DOI: 10.15514/ISPRAS-2019-31(4)-3.

[2]. POK kernel repository. Available from: https://github.com/pok-kernel/pok, accessed 10.11.2020.

[3]. Basic Spin Manual. Available from: http://spinroot.com/spin/Man/Manual.html, accessed 10.11.2020.

[4]. Delange J. Intégration de la sécurité et de la sûreté de fonctionnement dans la construction d'intergiciels critiques. THÈSE, Télécom ParisTech, 2010 (in French).

[5]. Delange J., Gilles O., Hugues J., and Pautet L. Model-based engineering for the development of ARINC653 architectures. SAE International Journal of Aerospace, vol. 3, issue 1, 2009, pp. 79-86.

[6]. Delange J. AADL in Practice: Become an expert in software architecture modeling and analysis. Reblochon Development Company, 2017, 252 p.

[7]. Specification ARINC. 653-2: Avionics application software standard interface: Part 1-required services. Technical report, Avionics Electronic Engineering Committee (ARINC), 2006.

[8]. Holzmann G. The model checker SPIN. In IEEE Transactions on software engineering, vol. 23, issue 5, 1997, pp. 279-295.

[9]. SPIN. Available from: https://github.com/nimble-code/Spin, accessed 10.11.2020.

[10]. Promela language reference. Available from: http://spinroot.com/spin/Man/promela.html, accessed 10.11.2020.

[11]. Старолетов С.М. Основы тестирования и верификации программного обеспечения. СПб., Лань, 2018 г., 344 стр. / Staroletov S. Basics of Verification and Testing. Spb., Lanbook, 2018, 344 p. (in Russuan).

[12]. Pnueli A. The temporal logic of programs. In Proc. of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977), 1977, pp. 46-57.

[13]. Hugues J., Delange J. Model-based design and automated validation of ARINC653 architectures using the AADL. In Cyber-Physical System Design from an Architecture Analysis Viewpoint, Springer, 2017, pp. 33-52.

[14]. Khoroshilov A. On formalization of operating systems behaviour verification. In Proc. of the International Conference on Computer Science and Information Technology (CSIT), 2017, pp. 168-172.

[15]. Khoroshilov A.V., Kuliamin V.V., Petrenko A.K. Verification of Operating System Components. System Informatics, issue 10, 2017, pp. 11-22.

[16]. Кулямин В.В., Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем. Труды ИСП РАН, том 28, вып. 3, 2016 г., стр. 189-208 / Kuliamin V.V., Lavrischeva E.M., Mutilin V.S., Petrenko A.K. Verification and analysis of variable

operating systems. Trudy ISP RAN/Proc.ISP RAS, vol.28, issue 3, 2016, pp. 189-208 (in Russian). DOI: 10.15514/ISPRAS-2016-1(2)-12.

[17]. Mallachiev K. M., Pakulin N. V., Khoroshilov A. V. Design and architecture of real-time operating system. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp. 181-192. DOI: 10.15514/ISPRAS-2016-28(2)-12.

[18]. Солоделов Ю.А., Горелиц Н.К. Сертифицируемая бортовая операционная система реального времени JetOS для российских проектов воздушных судов. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 171-178 / Solodelov Y. A., Gorelits N. K. Certifiable onboard real-time operation system JetOS for Russian aircrafts design Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 3, 2017, pp. 171-178. DOI: 10.15514/ISPRAS-2017-29(3)-10.

[19]. Klein G. et al. seL4: Formal verification of an OS kernel. In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, 2009, pp. 207-220.

[20]. Staroletov S., Shilov N. Applying Model Checking Approach with Floating Point Arithmetic for Verification of Air Collision Avoidance Maneuver Hybrid Model. Lecture Notes in Computer Science, vol. 11636, 2019, pp. 193-207.

[21]. Staroletov S. Partitioned OS Model Implementation. Available at: https://github.com/SergeyStaroletov/PromelaSamples/blob/master/Sched.pml, accessed 10.11.2020. DOI: 10.5281/zenodo.3757397.

[22]. Старолетов С. Модель многораздельной операционной системы для целей формальной верификации. Государственная регистрация программы для ЭВМ, no. 2020614016, 2020 / Staroletov S. Partitioned OS model for formal verification purposes. State registration of computer software, no 2020614016, 2020 (in Russian).

Information about the author / информация об авторе

Сергей Михайлович СТАРОЛЕТОВ – кандидат физико-математических наук, доцент кафедры прикладной математики. Сфера научных интересов: формальная верификация, model checking, киберфизические системы, операционные системы.

Sergey Mikhailovich STAROLETOV – Candidate of Physical-Mathematical Sciences (PhD), Associate Professor at the department of Applied Mathematics. Research interests: formal verification, model checking, cyber-physical systems, operating systems.

DOI: 10.15514/ISPRAS–2020–32(6)–5

О разработке Оберон-системы с заданными свойствами эргодичности

Д.В. Дагаев, ORCID: 0000-0003-0343-3912 <dvdagaev@oberon.org>
АО «Русатом – Автоматизированные системы управления»,
115230, Россия, г.Москва, Каширское шоссе, д. 3, корп. 2, стр. 16

Аннотация. Эргодичность систем характеризуется сохранением стационарности распределения состояний. Для компьютерных систем важно не допустить деградацию свойств системы со временем. Эргодичность особенно необходима для критически важных систем в ответственных отраслях. Разработка ПО на основе требований функциональной безопасности стандарта МЭК 60880 категории А реализуется только на вновь создаваемом ПО, отвечающим данным, наиболее жестким требованиям для АЭС, при невозможности использовать стандартные ОС и компиляторы. Для данных целей был реализован прототип среды исполнения и прикладного ПО дисплейной системы командного управления (ДСКУ). Среда исполнения (рантайм) создавалась на основе Active Oberon системы A2. A2 представляет собой однопользовательскую многозадачную систему. Область применения – промышленные встроенные системы реального времени, системы повышенной надежности. Среда исполнения ДСКУ реализована существенной переработкой минимального подмножества A2 для обеспечения требований стандарта. Система ограничений, формируемая по требованиям стандарта, дает возможность создавать компьютерные системы с новыми свойствами. Использование данных ограничений приводит к доказательству отсутствия возможности появления вызываемых ими сбоев и позволяет рассматривать компьютерную систему исходя из презумпции неэргодичности.

Ключевые слова: эргодичность; Оберон; надежность

Для цитирования: Дагаев Д.В. О разработке Оберон-системы с заданными свойствами эргодичности. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 67-78. DOI: 10.15514/ISPRAS–2020–32(6)–5

Благодарности: Международный общественный научно-образовательный проект Информатика-21.

Towards Developing of Oberon System with Specific Requirements of Ergodicity

D.V. Dagaev, ORCID: 0000-0003-0343-3912 < dvdagaev@oberon.org >
Rusatom – Automated Control Systems JSC,
Kashirskoye Shosse, 3/2/16, Moscow, Russian Federation, 115230

Abstract. The ergodic system keeps the time average is the same for almost all initial points. It is important for computer systems to prevent the degradation of the properties of the system over time. Ergodicity is especially required for mission-critical systems in demanding industries. Software development based on the functional safety requirements of the IEC 60880 category A standard is implemented only on newly created software that meets the most stringent requirements for nuclear power plants, it is impossible to use standard operating systems and compilers. For these purposes, a prototype of the runtime environment and application software of the command display system (DSCU) was implemented. The runtime was created based on the Active Oberon A2 system. A2 is a single-user multi-tasking system. Application area - industrial embedded real-time systems, high reliability systems. The DSKU execution environment is implemented by a significant revision of the minimum subset A2 to meet the requirements of the standard. The system of restrictions formed according to the requirements of the standard makes it possible to create computer systems with new properties. The use of

these constraints leads to the proof that there is no possibility of the occurrence of the failures they cause and allows us to consider a computer system based on the presumption of non-ergodicity. This «via negative» approach is based on restrictions, the addition of which allows one to obtain new qualitative properties. The more restrictions, the greater the gain in system reliability and stability.

Keywords: ergodicity; Oberon; reliability

For citation: Dagaev D.V. Towards developing of Oberon system with specific requirements of ergodicity. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 67-78 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–5

Acknowledgements. Informatika-21 (a non-commercial project to promulgate scientific rationality for IT education).

1. Введение

Мы постоянно сталкиваемся с разного рода примерами, когда функционирующая минуту назад компьютерная система неожиданно переходит в какой-то режим, при котором нормальная работа невозможна или, как минимум, затруднена. Вот произошел сбой, мы увидели «синий экран смерти», и в результате имеем ту же ОС, то же прикладное программное обеспечение, но в данном режиме изменилось важное – состояние. Пользователь вынужден вмешаться, нажать кнопки аппаратного рестарта и начать процесс функционирования заново. При работе с Windows часто происходят зависания, приводящие к деградации выполнения компьютером функций пользовательского интерфейса. Каждый такой случай имеет свои первопричины, но в конечном итоге пользователь определяет, что данное состояние с его точки зрения недопустимо, и перезагружает компьютер. Примеров таких множество, и особенно неприятно, когда это происходит с автономными устройствами: роутерами, ТВ-приставками, контроллерами АСУ ТП.

Независимо от первопричин такого поведения систем самое общее рассмотрение показывает, что в компьютерной системе со временем изменяются свойства, относящиеся к программному обеспечению. Это может быть используемые ресурсы, как динамическая память и процессорное время. Это могут быть ресурсы ПО, как количество открытых файлов или потоков. Изменение этих свойств определяется изменением состояния системы в целом. Плохо, если изменение состояния со временем приводит к изменению свойств в сторону ухудшения (например, меньше динамически запрашиваемой памяти).

Отсюда и вытекает рассмотрение эргодичности компьютерных систем как описания процесса изменения состояния и деградации свойств. Важны также и способы предотвращения деградации свойств.

Существуют разные системы требований, относящиеся к свойствам внедряемых компьютерных систем, для АЭС они определены в перечне отраслевых международных стандартов. Требования стандартов устанавливают конкретные ограничения, тогда как эргодичность определяет общие классифицируемые черты и описывает тенденции происходящих процессов.

В данной статье описываются проблемы деградации свойств и рассматриваются способы их решения на примере разработанной системы ДСКУ средствами Active Oberon на базе рантайма A2.

2. Свойство эргодичности применительно к программным системам

Когда мы говорим об эргодичности, имеется в виду стационарность, вероятности αi состояний системы не зависят от времени и не зависят от распределения вероятностей в начальный момент времени, то есть: $\alpha i = const$.

В работах [1, 2] показана важность свойства эргодичности, при невыполнении которого распределение состояний нестационарных систем будет существенно отличаться от

начальных состояний со временем. Авторы доказывают, что для таких систем среднее по времени и в количественном выражении, и в динамике, существенно отличаются. Это обусловлено тем, что для неэргодических систем распределения вероятностей начинают зависеть от предыстории, и случайная величина перестает быть независимой.

Свойство эргодичности применимо и к программным системам. Например, разработано серверное ПО и тестируется требования на безотказность в течении 100 часов работы. Рассматриваются два способа проверки:

- одно серверное приложение тестируется непрерывно в течении 100 часов (по времени);
- 100 приложений тестируются по 1 часу каждое в автономной среде (по ансамблю).

Если в разработанном серверном ПО есть утечки памяти, то со временем используемая память будет расти, а свойства системы (свободная память) – деградировать. В связи с этим фактором вероятность отказа со временем будет расти, система становится не эргодической, и вероятность отказа за 100 часов непрерывной работы будет больше вероятности отказа по ансамблю. С точки зрения тестировщика такой случай является абсолютно неравноценной заменой, и в требованиях обычно пишут 100 часов непрерывной работы.

В данной работе будет идти разговор об эргодичности применительно к общему состоянию системы и о конкретных свойствах (например, свободная память), поведение которых будет описываться как поведение свойств эргодичности.

В идеализированной программной системе постулируются допущения об обязательном успешном выделении памяти за минимальное время, отсутствии ограничений стека, реальном времени без блокировок и гонок и нулевой стоимости синхронизации, системе исключений, вызывающей безопасный рестарт, идеальной сети с отсутствием потерь, с нулевой латентностью и отсутствием переполнений счетчиков времени. В реальной системе обоснованность этих допущений требует доказательств, в том числе и архитектурного характера, из которых можно сделать вывод об эргодичности. Далее будут перечисляться допущения с большей степенью детализации.

Задачей разработчиков надежного ПО является получение предсказуемых свойств эргодичности. Если данные свойства находятся в допустимых рамках, это гарантирует отсутствие вызываемых ими программных сбоев. Идею профилактики вместо хирургии высказывал еще Н.И.Пирогов: «будущее принадлежит медицине предохранительной» [3].

3. Требования функциональной безопасности к ДСКУ

Разработка ПО на основе требований функциональной безопасности стандарта МЭК 60880 категории А [4] реализуется только на вновь создаваемом ПО, отвечающим наиболее жестким требованиям для АЭС, при невозможности использовать стандартные ОС и компиляторы. В части ОС необходимо, как минимум, следующее:

- *B.2cb:* следует избегать использования универсального операционного программного обеспечения (операционных систем);
- *B.2cc:* если операционная система необходима, то ее применение следует ограничить небольшим числом простых функций;
- *B.2cd:* операционная система должна содержать только необходимые функции.

В части реального времени и времени обработки требуется, как минимум, следующее:

- *B.2dd:* время прогона не должно существенно изменяться в результате изменения входных данных;
- *B.2de:* значение изменения времени прогона, которое может быть вызвано входными данными, должно быть документально оформлено;
- *B.2dg:* объем данных, считываемых в течение одного вычислительного цикла, должен быть постоянным.

В части работы с памятью требуется, как минимум, следующее:

- *B.3c:* содержимое памяти должно быть защищено или контролироваться;
- *B.3db:* следует проверять правильность передачи любого параметра, включая проверку типа параметров;
- *B.3dc:* при адресации массива следует проверять его границы.

В части работы с прерываниями требуется, как минимум, следующее:

- *B.2e:* необходимо ограничивать применение прерываний;
- *B.2ea*: прерывания могут использоваться, если они упрощают проект ПО и не делают верификацию чрезмерно сложной;
- *B.2ed*: если прерывания используются, то для непрерываемых частей необходимо иметь оценки максимального времени вычислений, чтобы можно было рассчитать максимальное время, в течение которого прерывание запрещено.

Требования данного стандарта носят конкретный характер, однако они хорошо подпадают под определение свойств эргодичности.

Для данного набора требований необходимо было реализовать прототип среды исполнения и прикладного ПО дисплейной системы командного управления (ДСКУ). Прикладное ПО должно было поддерживать функции как контроллера в АСУ ТП, так и как человеко-машинного интерфейса (ЧМИ) со средствами индивидуального управления задвижками и насосами, встраиваемый в стойки промышленного исполнения АСУ ТП.

4. Реализация на основе Active Oberon системы A2

4.1 Оберон-система A2

Научная школа Н. Вирта в лице Швейцарской высшей технической школы Цюриха (Swiss Federal Institute of Technology Zurich, ETH Zurich) и научно-образовательного проекта Информатика-21[1] развивает уже 50-летнюю историю эволюционирования языков программирования Алгол-Паскаль-Модула-Оберон; поэтому выбор для средств для реализации прототипа был не случайным.

Среда исполнения (рантайм) создавалась на основе разработанной для X86 версии Active Oberon-системы A2[2]. A2 представляет собой однопользовательскую многозадачную систему. Область применения – промышленные встроенные системы реального времени, системы повышенной надежности. Архитектуру A2 можно назвать наноядерной, т.к. A2 реализует рантайм «на голом железе» – «bare machine» [5]. При этом имеется минимальные модули рантайма – Objects, Machine, драйверы и механизмы переключения приоритетный режим – пользовательский режим.

Среда исполнения ДСКУ реализована путем существенной переработки минимального подмножества A2 для обеспечения требований стандарта. Исключены все функции, не являющиеся необходимыми. Исключены средства поддержки многоядерности и потенциально проблемные области рантайма.

4.2 Модель памяти A2 и управление памятью ДСКУ

При рассмотрении вопросов работы с памятью делается допущение #1 – представление о неограниченности памяти:

1) отсутствуют утечки памяти;

[1] http://www.inr.ac.ru/~info21/

[2] https://github.com/metacore/A2OS

2) динамическая память не фрагментирована;
3) запрашиваемая аллокируемая память может и будет выделена;
4) время срабатывания сборщика мусора равно нулю;
5) пределы выделения стека отсутствуют;
6) своппинг срабатывает идеально за нулевое время;
7) постраничная организация памяти не оказывает воздействия на временные характеристика системы;
8) статическая память гарантированно выделяется при рестарте компонентов системы.

Модель памяти A2 основана на стандартных механизмах сегментации. При этом динамическая виртуальная память отображается на физическую с сохранением адресов [5], исключая области нулевых адресов, стека и специальных устройств. Таким образом, при обращении к нулевому указателю или к стеку за пределами адресуемой памяти формируется ошибка сегментации. Авторы A2 называют обращение к виртуальной памяти «ложной абстракцией» [5], при которой подкачка страниц по запросу не осуществляется.

Модель памяти A2 гарантирует правильность допущений #1.6 (отсутствие своппинга) и #1.7 (отсутствие постраничной организации памяти).

В системе ДСКУ реализован запрет управления памятью после фазы инициализации. Цикл работы системы разбит на 6 этапов:

1) загрузка системы;
2) фаза инициализации: объектам и данным предоставляется динамическая память; происходит выделение запрашиваемой стековой памяти (рекурсия отсутствует);
3) запрет выделения памяти, отключение сборщика мусора, запрет дисковых операций;
4) код старта активных объектов;
5) цикл обновления по таймеру;
6) окончание работы и перезагрузка.

После этапа 3 все операции по выделению динамической памяти прекращаются. Запрет выделения памяти означает установку программного исключения при каждом возможном обращении. На уровне рантайма нормальная работа происходит на этапе 5. Восстанавливаемые прерывания приводят к коду рестарта этапа 4 с переходом на 5. Фатальные ошибки приводят к этапу 6 – перезагрузке системы.

Данный механизм доказывает справедливость допущений #1.1-#1.5, #1.8 на этапах, начиная с 4. Если фаза инициализации пройдена, корректная работа с памятью становится доказанной. Система функционирует с фиксированным объемом динамической и стековой памяти, которые были выделены на этапе инициализации.

4.3 Реальное время и активные объекты

Реальное время и синхронизация соответствуют допущению #2:

1) отсутствуют синхронизационные тупики (deadlock);
2) нет гонок; вызванные ими искажения критически важных данных отсутствуют;
3) время прохождение через систему известно с фиксированной точностью;
4) непосредственное взаимное влияние процессов (активностей) друг на друга отсутствует;
5) прерывания системы ввода/вывода не оказывают влияния на процессы обработки.

Многозадачность в ДСКУ основана на активных объектах A2. Активный объект – это специальная концепция Active Oberon, при которой процесс инкапсулирован в объекте [5]. Активные объекты ДСКУ создаются на фазе инициализации, имеют время жизни, равное времени жизни ДСКУ и включают в себя процедуры старта и обновления.

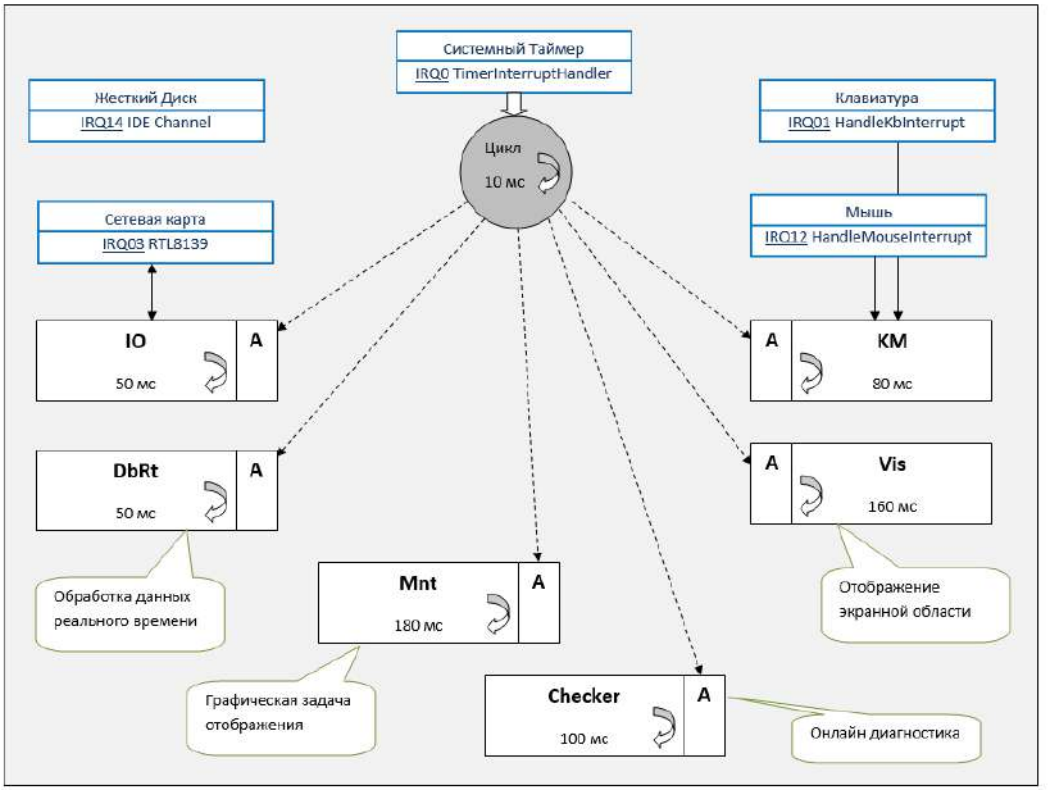

Рис. 1. Структура активных объектов ДСКУ
Fig. 1. Active Objects structure of DSKU

ДСКУ в исполнении ЧМИ состоит из следующих задач, реализуемых активными объектами (рис. 1):

- IO – ввод/вывод (в прототипе реализован как UDP с драйвером стека A2, во внедряемом варианте предполагалось использование полевой сети);
- DbRt – технологическая БД реального времени, представляющая собой массив структур статических и динамических параметров сигналов;
- KM – клавиатура и манипулятор (трекбол с заменой на touch screen во внедряемом варианте);
- Checker – онлайн диагностика состояния задач и целостности данных;
- Mnt – графическая задача отображения видеокадров и формирования команд оператора;
- Vis – формирование растра отображения экранной области.

Активные объекты изолированы, они взаимодействуют друг с другом только через объекты агентов (доказательство #2.4). Агенты осуществляют временную синхронизацию. Синхронизация обеспечивает привязку объекта к определенному фрейму вызова.

В части взаимодействия между объектами существенно, что согласно стандарту время прогона должно быть фиксированным (B.2dd, #2.3), и это должно объясняться. Поэтому взаимодействие между агентами объектов необходимо было построить по неблокирующей схеме. Это гарантирует, что объект обновляется только по временному циклу, и не ожидает наступления каких-то логических условий. Доказываются допущения #2.1, #2.2. Был выбран обмен по схеме двойной буферизации [6] (рис. 2).

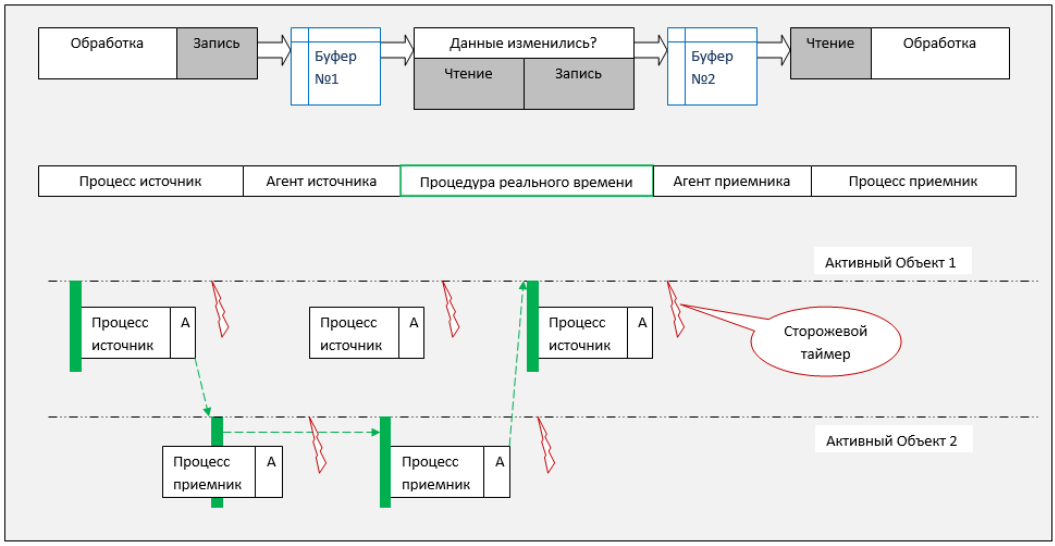

Рис. 2. Межзадачный обмен по схеме двойной буферизации
Fig. 2. Double buffering inter-task communication

Двойной буфер обеспечивает атомарный обмен области чтения на область записи. Это осуществляется процедурой реального времени, во время которой прерывания потока выполнения другими задачами запрещено.

Процессы большей длительности, например, обновление графического экрана, запускаются как задачи, разбиваемые при выполнении. При превышении времени счета задачи на 20% от установленного срабатывает сторожевой таймер, который устанавливает флаги овертайма.

Прерывания ввода/вывода ограничены по числу срабатываний за несколько фреймов вызова. При таком поведении заметна неравномерность движения трекбола. Это обеспечивает приемлемую работу манипулятора и клавиатуры. Дисковые операции запрещались после фазы инициализации (#2.5).

4.4 Исключения и восстановление

Исключения и восстановление соответствуют допущению #3:

1) исключения не приводят к потере ресурсов;
2) обработка исключения не приводит к генерации исключения внутри обработчика;
3) восстановление после обработки исключения всегда завершается успешно;
4) восстановление приводит к работоспособному состоянию, близкому к состоянию первого старта;
5) объект с исключением не оказывает влияние на работоспособность других объектов.

Активные объекты содержат процессы внутри себя. Конкретные объекты наследуются от базового объекта Task (рис. 3) и становятся активными. Система A2 обеспечивает инициализацию в фазе 2 вызовом конструктора &Init (…), при этом осуществляется аллокирование данных и связывание объектов с агентами. Начиная с фазы 4 происходит выполнение тела BEGIN {ACTIVE, SAFE} активного объекта в контексте процесса активного объекта, начиная с кода старта InitData. Код старта осуществляет мягкую инициализацию установкой начальных значений данных и далее по телу цикла следует фаза 5.

В фазе 5 реализован бесконечный цикл обновления данных Update по таймеру. При некорректных условиях работы программы генерируются программные исключения, относящиеся к данной задаче.

```
Task* = OBJECT
        VAR agent-: Agent; tAdvance, tBefore: INTEGER;
                statrec-: Dc.Statrec; dead: BOOLEAN; ticks: LONGINT;

        PROCEDURE StartAgent* (tAdvance, tBefore: INTEGER);
        BEGIN
                …
        END StartAgent;

        PROCEDURE Start*;
        BEGIN
                …
        END Start;

        PROCEDURE Finish*;
        BEGIN
                …
        END Finish;

        PROCEDURE &Init* (agent: Agent);
        BEGIN
                ASSERT(agent # NIL, 21);
                SELF.agent := agent; statrec.errcount := -1;
        END Init;

        PROCEDURE InitData;
        BEGIN
                agent.InitData; tAdvance := 0; tBefore := 0;
                …
                Start
        END InitData;

        PROCEDURE NextTsp*;
        BEGIN {EXCLUSIVE}
        END NextTsp;

        PROCEDURE Update*;
        BEGIN
        END Update;

BEGIN {ACTIVE, SAFE}
        IF fin THEN …
        ELSE
            InitData;
            LOOP
                BEGIN {EXCLUSIVE}
                        AWAIT(fin OR (agent.tsp >= agent.start));
                        IF fin THEN dead := TRUE; EXIT END;
                END;
                …
                Update;
                …
                IF ~fin THEN
                        CASE agent.number OF
                        | 1:
                                ASSERT(agent.status # agentError, 122)
                        …
                        ELSE
                                ASSERT(agent.status # agentError, 22)
                        END
                END
            END
        END
END Task;
```

Рис. 3. Фрагмент кода базового объекта Task
Fig. 3. Code snippet of the base Task object

Семантическая конструкция {ACTIVE, SAFE} уведомляет, что данная задача имеет безопасный рестарт. При возникновении исключения при обновлении/анализе данных тело активного объекта перехватывает трап и переходит к точке рестарта.

Точка рестарта – это начало тела объекта, первый оператор которого будет код старта InitData (рис. 4). Выполняется фаза 4, за ней идет переход к фазе 5 с бесконечным циклом обновления данных. Конфигурация памяти активного объекта не меняются, происходит только смена начальных условий в InitData. Такое архитектурное решение подтверждает

корректность допущений #3.1, #3.4. Следует заметить, что данные активного объекта изолированы от других объектов и не меняются, что подтверждает доказанность независимости от других #3.5.

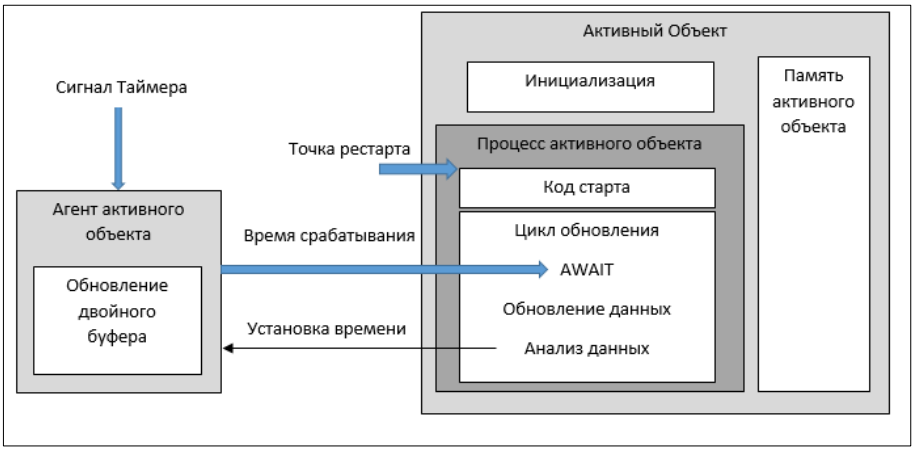

Рис. 4. Рестарт активного объекта
Fig. 4. Active Object restart

Обработчик исключений в ДСКУ не имеет внутреннего состояния и не создает новые данные. Он осуществляет обновление массивов статистики отказов для данной задаче. Таким образом, #3.2 (нет генерации исключения внутри обработчика) выполняется, а #3.3 выполняется с оговоркой, что не восстанавливаемые исключения приводят к перезагрузке системы.

В части оценки эргодичности системы при периодических восстановлениях системы (например, вызванных овертаймами) можно утверждать, что каждое новое восстановленное состояние идентично предыдущему с точностью до временных меток.

4.5 Отсутствие потерь информации при сетевых сбоях

В качестве допущений #4 корректной работы сети используются сформулированные в [7]:

1) сетевая передача является надежной;
2) латентность равна 0 или не учитывается;
3) пропускная способность бесконечна;
4) транспортные издержки равны 0 или не учитываются.

Согласно стандарту 60800, «*B.2dg:* объем данных, считываемых в течение одного вычислительного цикла, должен быть постоянным». Поэтому по сети осуществляется циклическая передача всех данных фиксированного размера с циклом обновления 10-50 мс. При этом передача осуществляется по дублированной сети. Даже при наличии многократных сетевых сбоев информация приходит в ДСКУ со следующим циклом, т.е. с некоторой задержкой, кратной периоду обмена.

Для прототипа использовался протокол UDP. Использование «эластичных» протоколов, в частности, TCP ухудшает свойства эргодичности при сетевой передаче, т.к. соединение может рваться и не восстанавливаться при сложных условиях эксплуатации.

Циклический прием данных обеспечивает надежное #4.1 гарантированное считывание порции данных за 2 сетевых цикла, при условии, что произошло не более 1 сбоя хотя бы в одной из 2 сетевых карт. Латентность #4.2 и транспортные издержки #4.4 не учитываются, т.к. ДСКУ не предназначена для работы через радиоканалы. Проблемы с пропускной способностью #4.3 тоже нет, т.к. при постоянной передаче фиксированного размера

максимальная пропускная способность является номинальной. И на этапе отладки с номинальным потоком сигналов пропускная способность легко тестируется.

4.6 Отсутствие переполнений времени и проблемы 2038

Проблемы с переполнением времени соответствуют допущению #5:

1) отсутствуют переполнения счетчиков времени;
2) величина будущей временной метки всегда больше текущей;
3) нет скрытых блокировок ожидания.

Существует множество известных проблем переполнения времени (проблемы 2038, 2106). Они связаны с переходом через 0 целочисленного счетчика времени. Последствия такой неэргодичности могут быть катастрофическими, например, приложения, использующие функцию Linux nanosleep, для многих ядер зависают при переходе 19 января 2038 года.

Единственный таймер ДСКУ не привязан к текущему времени или временным зонам и обновляет 64-разрядный счетчик tsp с момента начала загрузки, а не начального момента времени 1900 или более раннего года. Утверждения #5.1, #5.2 доказаны, т.к. нет и не будет оборудования, способного непрерывно отработать 10^8 лет при цикле 1 мс. В системе нет понятия текущего времени. Задержки реализуются только обратным отсчетом установленных начальных значений. Такая схема гарантирует отсутствие переполнений (проблемы 2038, 2106). Системы синхронизации и корректировки времени типа NTP не используются ДСКУ, и в целом для категории А. Следовательно, блокировок ожидания #5.3 нет.

5. Недостатки предлагаемого подхода

Повышение надежности и переход системы в стабильно стационарное состояние имеет свою обратную сторону. Процессорное время будет использовано гораздо менее эффективно. Выполнение требований стандарта исключает циклы вида WHILE DO, REPEAT UNTIL, которые не имеют постоянного числа итераций. Даже алгоритм линейного поиска выполняется циклом FOR без возможности преждевременного завершения при успешном окончании. Входные события, например, изменение состояний дискретного сигнала, обрабатываются не как события, а как весь массив входных данных.

Увеличение интенсивности прерываний наблюдалось при замене трекбола на манипулятор-мышь, при интенсивной работе которое наблюдались рывки. Это объяснялось ограничением числа прерываний за цикл обмена задачи.

Следует заметить, что применение данных систем необходимо при реализации простых алгоритмов в части защит, блокировок, управления АСУ ТП. Более сложные алгоритмы не имеют таких требований в части функциональной безопасности.

С точки зрения аппаратного обеспечения работа с постоянной стабильной нагрузкой на процессор будет даже предпочтительным решением, а современные мощности позволят все и далее переходить ко все более надежным решениям.

6. Презумпция неэргодичности

Вопрос наличия свойства эргодичности возникает для многих систем, предназначенных к долгосрочной эксплуатации. В большинстве случаев разработчики не проектируют заранее систему с учетом указанных свойств, а реагируют на возникновении конкретных технических проблем, пока это не станет носить системный характер. При этом программная система постулируется работоспособной, если она успешно функционирует сразу после загрузки.

Презумпция неэргодичности предполагает, что данное свойство отсутствует, и разработчику необходимо обосновать эргодичность системы, в том числе и рассмотрением архитектурных особенностей. Оберон-системы на основе A2 могут стать хорошим фундаментом для построения. Представленная система ДСКУ обладает доказательствами эргодичности в части:

- управления памятью;
- обеспечения реального времени;
- механизма обработки исключений;
- сетевого обмена данными;
- обеспечения управления временем.

Данный список может быть расширен при рассмотрении требований конкретных промышленных стандартов, которые не принимались во внимание, например, Авионики. Но тенденция сохраняется.

7. Заключение: развитие от отрицания «via negativa»

Знание, от каких технологических решений следует отказаться, часто более ценно, чем знание, какие технологические решения следует применять. Механизмы, приводящие к улучшению свойств эргодичности, основаны на ограничениях и запретах:

1) предсказуемость работы с памятью обеспечивается отказом от виртуальной подкачки страниц в пользу более простой прямой адресации;
2) гарантии управления памятью основаны на запрете динамического выделения после фазы инициализации;
3) жесткое реальное реализуется отказом от ОС на «голом железе»;
4) межзадачный обмен осуществляется только одним допустимым способом через двойную буферизацию. Другие способы исключены;
5) при рестарте запрещена инициализация, только программный код старта;
6) при сетевой передаче допускается только обмен посылками фиксированной длины;
7) установка времени и синхронизации времени запрещена.

Обобщая эти факторы, можно сделать вывод, что отказ от большинства используемых технологий приводит совершенно новым качествам, недостижимым путем добавления каких-либо еще более изощренных программных решений.

И, что важно, построенная на таких аскетических принципах программная система не только обладает свойствами эргодичности – она также становится объяснимой и доказуемой.

Список литературы / References

[1]. O. Peters, M. Gell-Mann. Evaluating gamples using dynamics. Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 26, issue 2, 2016, article id 023103.
[2]. O. Peters. The ergodicity problem in economics. Nature Physics, vol. 15, issue 12, 2019, pp. 1216-1221.
[3]. Соловьева А.А., Цапкова Н.Н., Покровский В.И. Будущее принадлежит медицине предохранительной – Н.И.Пирогов. Терапевтический архив, том 83, no. 11, 2011 г., стр. 5-9 / Solovyeva A.A., Tsapkova N.N., Pokrovsky V.I. "Medicine of the future is preventive medicine" N.I. Pirogov. Therapeutic archive, vol. 83, no. 11, 2011, pp. 5-9 (in Russian).
[4]. ГОСТ Р МЭК 60880, Программное обеспечение компьютерных систем, выполняющих функции категории А, 2009 / GOST R IEC 60880, Software for computer systems performing category A functions, 2009 (in Russian).
[5]. Pieter J. Muller, The Active Object System Design and Multiprocessor Implementation. Diss. ETH No. 14755, for the degree of Doctor of Technical Sciences, ETH Zurich 2002, 197 p.

[6]. S. Louise, M. Lemerre, C. Aussagues and V. David. The OASIS Kernel: A Framework for High Dependability Real-Time Systems. In Proc. of the IEEE 13th International Symposium on High-Assurance Systems Engineering, 2011, pp. 95-103.
[7]. Arnon Rotern-Gal-Oz. Fallacies of Distributed Computing Explained. URL: https://www.researchgate.net/publication/322500050_Fallacies_of_Distributed_Computing_Explained, accessed 20.11.2020.

Информация об авторе / Information about the author

Дмитрий Викторович ДАГАЕВ – главный эксперт «Русатом – Автоматизированные системы управления», консультант проекта «Информатика-21». Сфера научных интересов: разработка системного ПО АСУТП, разработка кросс-платформенных Оберон-технологий.

Dmitry Victorovich DAGAEV – Chief Expert of Rusatom – Automated Control Systems JSC, consultant of the Informatika-21 project. Research interests: DCS system software development, cross-platform Oberon technologies investigation.

DOI: 10.15514/ISPRAS–2020–32(6)–6

Проектирование высоконагруженных систем

В.А. Рудометкин, ORCID: 0000-0003-2718-7373 <vasiliy.rudometkin@gmail.com>
ООО СТРИМ,
115470, Россия, г. Москва, Проектируемый проезд № 4062, д. 6 стр. 2

Аннотация. В настоящее время большинство сервисов переходят в онлайн, что позволяет пользователям получать услугу в любое время. Высокая доступность услуги ведет к росту количества пользователей, что влечет за собой повышение нагрузки на систему. Высокая нагрузка оказывает негативное влияние на компоненты системы, что может привести к сбоям функционирования и потери данных. В статье рассмотрено несколько подходов к проектированию и мониторингу, следование которым поможет предотвратить неправильное функционирование системы. Описан наиболее популярный способ распределения области ответственности каждого сервиса, в соответствии с паттерном DDD, применение которого позволит разделить компоненты системы логически по использованию и физически при масштабировании системы. Данных подход будет полезен также и при масштабировании команды, позволяя разработчикам независимо работать над разными компонентами системы, не мешая друг другу. Интеграция новых людей в проект также будет занимать кратчайшие сроки. При проектировании архитектуры системы стоит уделить внимание и схеме взаимодействия сервисов между собой. Использование паттерна CQRS позволяет разнести чтение и запись в разные компоненты, что в дальнейшем позволяет пользователю быстро получать ответ от системы. Особое внимание в статье уделено мониторингу системы, так как при увеличении размера системы время поиска ошибок в системе занимает большое время, приводя к долгой недоступности системы, что может повлечь за собой потерю клиентов. Все описанные в статье способы применены на многих проектах, например, МТС ПОИСК. Благодаря правильно спроектированной системе удалось сократить время ожидание ответа сервиса с двух минут до нескольких секунд без потери качества результата, а сложная система мониторинга системы позволяет в режиме реального времени отслеживать все процессы внутри системы и предотвращать аварии. В итоге, в начале проектирования системы следует особое внимание уделить архитектуре, вопросу мониторинга и тестирования системы. Впоследствии эти временные вложения позволят снизить риски потери данных и недоступности работы системы.

Ключевые слова: высоконагруженная система; DDD; REST; сервера очередей; socket; ELK; CQRS; МТС ПОИСК, проектирование

Для цитирования: Рудометкин В. А. Проектирование высоконагруженных систем. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 79-86. DOI: 10.15514/ISPRAS–2020–32(6)–6

Designing Highly Loaded Systems

V.A. Rudometkin, ORCID: 0000-0003-2718-7373 <vasiliy.rudometkin@gmail.com>
STREAM LLC,
6c2, Projected Drive 4062, Moscow, 115470, Russia

Abstract. Nowadays, most of the services are moving online, which allows users to receive the service at any time. The high availability of the service leads to an increase in the number of users, which entails an increase in the load on the system. High load has a negative impact on system components, which can lead to malfunctions and data loss. To avoid this, the article discusses several design and monitoring approaches, the observance of which will help prevent system malfunctioning. The article describes the most popular way to distribute the area of responsibility of each service, in accordance with the DDD pattern, the use of which will allow you to separate the components of the system logically by use and physically when scaling the system.

This approach will also be useful when scaling a team and allow developers to work independently on different system components without interfering with each other. The integration of new people into the project will also take the shortest possible time. When designing the system architecture, it is worth paying attention to the scheme of interaction between services. Using the CQRS pattern allows you to separate reading and writing into different components, which later allows the user to quickly receive a response from the system. Particular attention in the article is paid to monitoring the system, since with an increase in the size of the system, the time to search for errors in the system reaches a large amount of time, which can lead to a long unavailability of the system, which will entail the loss of clients. All the methods described in the article have been applied on many projects, for example, MTS POISK. Thanks to a properly designed system, it was possible to reduce the waiting time for a service response from two minutes to several seconds without losing the quality of the result, and a sophisticated system monitoring system allows you to monitor all processes within the system in real time and prevent accidents. As a result, at the beginning of the system design, special attention should be paid to the architecture, the issue of monitoring and testing the system. Subsequently, these temporary investments will reduce the risks of data loss and system unavailability.

Keywords: highly loaded system; DDD; REST; queue server; socket; ELK; CQRS; MTS SEARCH, system design

For citation: Rudometkin V.A. Designing highly loaded systems. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 79-86 (in Russian). DOI: 10.15514/ISPRAS-2020-32(6)-6

1. Введение

В 21-м веке все больше сервисов переходят в онлайн, позволяя выполнять большинство каждодневных операций, не выходя из квартиры. Сегодня можно заказать еду, уборку, доставку одежды, найти работу и даже купить машину или квартиру, сидя за компьютером. Эти сервисы становятся с каждым днем сложнее, охватывая все больший спектр услуг, привлекая большое количество пользователей, что приводит к повышению нагрузки на систему. Высокая нагрузка на систему приводит к долгой обработке запросов пользователей, повышает процент потери данных и отказа системы, вследствие чего клиент выберет другого поставщика услуг. Для решения этой проблемы необходимо особое внимание уделить проектированию архитектуры системы.

Высоконагруженной системой называется система, нагрузку которой не может выдержать один сервер или у которой имеются тысячи или миллионы пользователей [1]. При проектировании новой архитектуры необходимо выбрать способ управления данными, способ ограничения контекста компонентов системы и способ их взаимодействия.

В статье рассмотрено несколько подходов к проектированию и мониторингу, соблюдение которых поможет предотвратить неправильное функционирование системы. Описан наиболее популярный способ распределения области ответственности каждого сервиса, в соответствии с паттерном DDD, применение которого позволит разделить компоненты системы логически по использованию и физически при масштабировании системы.

Использование паттерна CQRS позволяет разнести чтение и запись в разные компоненты, что в дальнейшем позволяет пользователю быстро получать ответ от системы. Особое внимание в статье уделено мониторингу системы, так как при увеличении размера системы время поиска ошибок в системе достигает большого количества времени, что может привести к долгой недоступности системы, которое повлечет за собой потерю клиентов.

2. Родственные работы

В [2] в общих чертах рассмотрен паттерн DDD и кеширование данных, которые меняются редко, что в высоконагруженной системе бывает крайне редко. В этой статье не рассматриваются способы кеширования данных, но подробно раскрывается способы правильного использования паттерна DDD.

В статье [3] приведен пример использования паттерна CQRS и доказана эффективность такого подхода, но не рассмотрена проблема интеграции паттерна в высоконагруженную систему. Этот вопрос необходимо рассматривать при проектировании системы, так как

неправильная интеграция в существующую систему может значительно усложнить поддержку системы, а внедрение может занять продолжительное время.

В работе [4] подробно описано проектирование систем с использованием паттернов и различных подходов, но паттерн CQRS отделен от событий. В настоящей статье предлагается рассмотреть совмещения этих двух подходов, что принесет дополнительное повышение производительности и снижение нагрузку на систему.

3. Управление данными

Архитектуру системы необходимо проектировать, опираясь на основные данные в системе, например, при проектировании файлового хранилища стоит определить, как будут храниться файлы – на диске с доступом по протоколу FTP или в базе данных.

Наиболее удобным способом хранения файлов является хранение ссылок на файлы в базе данных. При реализации такого решения в базе данных будут находиться идентификаторы файлов и их метаданные. На дисках или в облачных хранилищах будут храниться сами файлы. Для использования, у пользователя или другого ресурсах, которым необходим файл, хранится только идентификатор, по которому можно получить как файл, так и всю связанную с ним информацию. Основное преимущество такого подхода в том, что не важно, где хранить файл – или в базе данных в бинарном виде, например, или на диске, или в облачном хранилище.

Использовать базы данных не рекомендуется для хранения файлов, так как база данных из-за хранения файлов начинает занимать в разы больше дискового пространства, что приводит к долгому процессу создания бекапов и их восстановления.

Для задач, в которых структура данных недостаточно хорошо известна, подойдут NoSQL базы данных [5]. Основные типы нереляционных баз данных представлены на рис. 1.

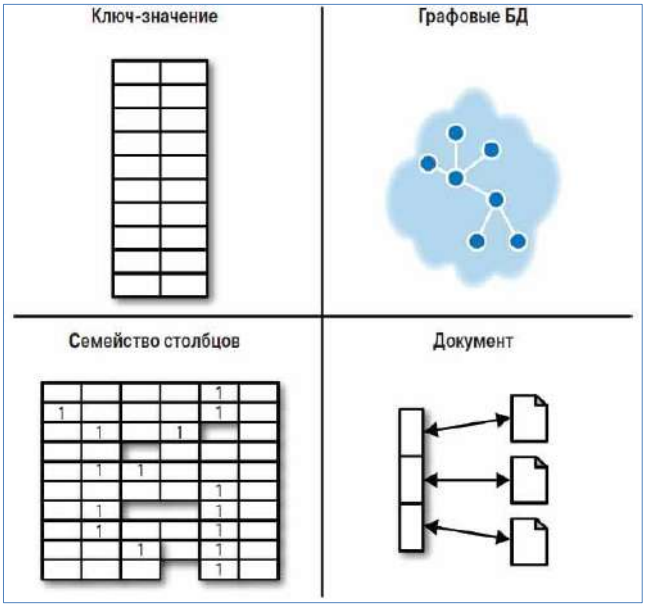

Рис. 1 Секторы NoSQL-хранилищ [6]
Fic.1 NoSQL storage sectors [6]

- *Ключ-значение* – такой тип нереляционной базы данных будет полезен, когда необходимо получить объект по идентификатору. Например, это может быть кеширование данных – с помощью уникального ключа можно быстро получать любые данные. Реализацией такого подхода может служить, например, база данных Redis [7].

- *Графовые системы* – данная структура будет полезна, если в системе используются связи между объектами одной коллекции, например, для получения корректного трека *n* точек GPS точек должны идти последовательно. Примером является база данных Arango [8].
- *Семейство столбцов* – отлично походит для метрик, когда есть большой поток данных от сервиса, а отображать его необходимо на разных графиках, удобно делать выборку по столбцам и отображать необходимые данные. Пример – ClickHouse [9].
- *Документ* – данный тип нереляционной базы данных позволяет хранить динамическую структуру значения. Такой тип базы данных будет полезен для структур данных, в которых невозможно предсказать полный набор полей. Пример – MongoDB [10].

Для гарантии целостности данных подойдут реляционные базы данных, такие как PostgreSQL, MS SQL Server, MySQL и другие. Основное, на что необходимо обратить внимание, – индексирование таблиц для быстрого доступа к данным по их уникальным идентификаторам. Необходимо позаботится о шардировании базы данных для повышения отказоустойчивости и о репликации для гарантии сохранности данных. При выборе реляционной базы данных также следует учитывать их особенности; например, база данных PostgreSQL [11] позволяет хранить тип данных JSONB – двоичную разновидность формата JSON, у которой пробелы удаляются, сортировка объектов не сохраняется, вместо этого они хранятся оптимальным образом и сохраняется только последнее значение для ключей-дубликатов.

4. Паттерн DDD

При разработке сервисов нового приложения часто придерживаются монолитной архитектуры, при которой один сервис обрабатывает практически любые задачи – чтение и запись в базу данных, бизнес-логика. Такой поход имеет свои преимущества при старте проекта – экономится время и ресурсы на первый запуск приложения, но развитие такой системы затруднено из-за невозможности масштабирования. Для решения этой проблемы разработана микросервисная архитектура, которая позволяет разделить монолит на небольшие отдельные компоненты. На этапе проектирования микросервисов появляется проблема разграничения логики каждого сервиса, так как если определить неправильный контекст сервиса, то выигрыша от такого подхода не будет. Наиболее эффективным способом является разграничение области ответственности сервиса в соответствии с бизнес-моделью; например, один сервис обрабатывает все потоки данных, которые касаются персональных данных, другой поддерживает транзакции пользователей и т.д. Такой подход описан в наборе принципов Domain Driven Design (DDD) [12].

Основными принципами, которыми оперирует DDD являются следующие.

1) *Ограниченные связи.* Так как сервис работает в ограниченном контексте, то и связей будет минимальное количество.
2) *Целостность.* При любом исходе в базе данных не останется несогласованных данных. В случае, если происходит ошибка распределенных транзакций, и данные портятся, проблема решается на другом уровне.
3) *Взаимосвязь.* При проектировании системы по методу DDD получится четкая структура, в которой каждый отдельный компонент имеет связанные, но независимые элементы. Это значит, что данные в одном сервисе зависят от данных в другом сервисе, например, пользователь и его счет, но изменение одной из сущностей не потребует изменения другой.

Для максимальной эффективности такого подхода базы данных должны быть также изолированы друг от друга – одна база на один сервис. Такое решение позволит сократить время на восстановление базы данных и снизить нагрузку на сервер.

5. Взаимодействие компонентов и паттерн CQRS

При реализации приложения с архитектурой по DDD паттерну приводит к большому количеству сетевых взаимодействий между компонентами системы, поэтому стоит выбрать способы взаимодействия каждого компонента системы.

Выбор способа взаимодействия между компонентами зависит от нескольких факторов:

1) количество и частота передаваемых данных;
2) ответная реакция на вызов компонента.

В первом случае, если необходимо передать/получать большое количество данных, или если вызов компонента осуществляется часто, то стоит выбрать способ непрерывного взаимодействия – *socket*. При таком подходе открывается непрерывное взаимодействие между двумя компонентами на серверной части приложения или клиент-серверном взаимодействии, и осуществляется непрерывная передача данных. При реализации взаимодействия через *socket* не требуется ответная реакция вызываемого компонента. Один из недостатков данного решения состоит в том, что поскольку под каждое соединение открывается свой порт, количество соединений для одного серверного компонента ограничено величиной 65535, но при таком подходе сервис может не успевать обрабатывать все входящие данные, поэтому стоит ориентироваться на сложность обрабатываемых задач.

В случае, когда взаимодействий не так много и передаваемых данных меньше, можно использовать более простую синхронную реализацию взаимодействия компонентов и клиента с серверов – *rest*.

При проектировании архитектуры системы, если ответ не нужен сразу или необходимо провести изменения в большом количестве компонентов, то стоит использовать событийную модель – отправлять событие в сервер очередей, используя, например, *kafka* [13] и другие сервисы. Слушатели должны подписаться на это событие и провести необходимые операции. Основное преимущество данного подхода – минимизация нагрузки на систему, так как слушатель прочитает столько сообщений, сколько может обработать, возможность провести изменения в большом количестве компонентов практически одновременно. Основной недостаток такого подхода – невозможна простая реализация, при которой можно вернуть ответ клиенту, возникают дополнительные точки отказа и появляется нагрузка на поддержку сервера очередей.

Для получения результата обработки события клиентом следует решить следующую проблему: компонент, который получил данные, занят их обработкой, и дополнительные запросы на выдачу информации приведут к созданию дополнительных соединений, лишней нагрузки на компонент и базу данных. Поэтому процессы добавления данных и их выдачи необходимо разделить – создать разные компоненты, один из которых будет принимать данные от клиента, а второй их отдавать.

Соответствующий паттерн проектирования называется CQRS (Command and Query Responsibility Segregation, разделение ответственности команд и данных) [14]. Этот паттерн принято кратко характеризовать следующим образом: метод должен быть либо командой, выполняющей какое-то действие, либо запросом, возвращающим данные, но не одновременно тем и другим. В основе CQRS лежит принцип CQS (Command and Query Segregation, разделение команд и запросов), введенный Бертраном Мейером (Bertran Meyer) в его книге [15].

Принцип императивного программирования позволяет снизить нагрузку на базу данных и разделить потоки чтения и записи данных. В результате система гарантирует быстрый ответ пользователю на получение данных, а запись их не затрагивает работу пользователя за счет выполнения операций чтения и записи над разными базами данных (рис. 2).

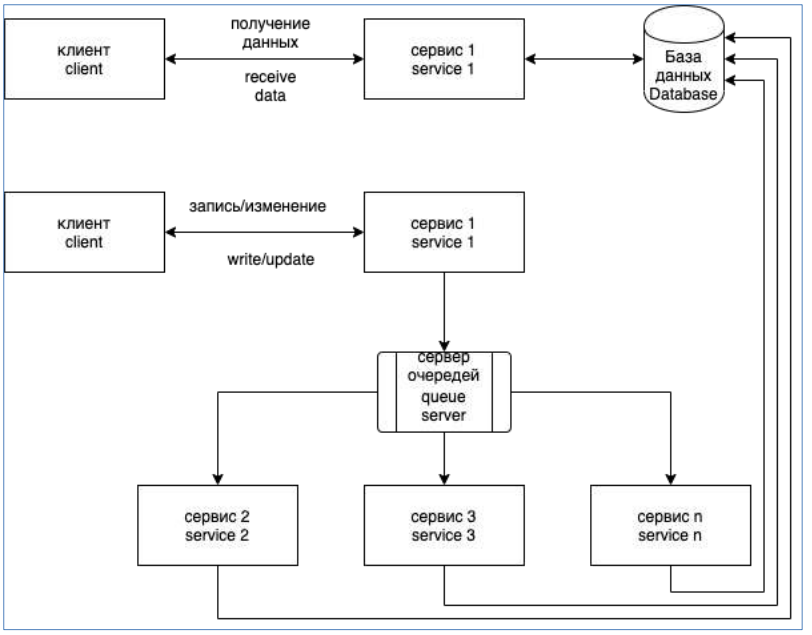

Рис. 2. Схема взаимодействия сервисов в высоконагруженной системе
Fig. 2. Service interaction scheme in a highly loaded system

6. Примеры практического применения паттернов

Описанные в статье способы применены во многих проектах, например, МТС ПОИСК [16], открытие нового счета в Альфа-Банке и т.д.

Приложение МТС Поиск – сервис для поиска людей, основанных на геоданных. Трек пользователя представляет из себя набор связанных точек – граф, поэтому применение графовой нереляционной базы данных позволило хранить маршруты пользователя в базе данных и быстро получать его. Для этого решения была выбрана графовая NoSQL база данных Arango.

Получение в приложении некоторых сущностей, которые меняются редко, занимало много времени, поэтому была интегрирована нереляционная база данных «ключ-значение» Redis для хранения готовых объектов. Время получения сущностей по уникальным идентификаторам удалось сократить в разы, также удалось уменьшить количество сетевых взаимодействий, нагрузку на базу данных и другие компоненты системы, которые задействованы в процессе сбора данных.

В проекте Новый счет для юридических лиц в компании Альфа-Банк необходимо было обрабатывать большое количество файлов и разное количество метаданных, связанных с каждым из них, поэтому было выбрано NoSQL решение MongoDB. В результате удалось решить проблему хранения файлов и всей связанной с ними информации, хранить разные форматы данных, так как в некоторых случаях метаданных может быть значительно больше. За счет шардирования удалось легко масштабировать базу данных и получить хорошую производительность.

Паттерн CQRS применялся в приложении МТС Поиск. Получение всех новых данных выведено в отдельные процессы, которые работают отдельно от компонентов, которые выдают информацию пользователю. За счет этого получилось упростить разработку приложения.

7. Заключение

При проектировании высоконагруженной отказоустойчивой системы с возможностью как вертикального, так и горизонтального масштабирования следует разделять чтение и запись в системе, разрабатывая архитектуру по паттерну CQRS и соблюдая правила разделения сервисов по паттерну DDD. Требуется выбрать подходящую базу данных, возможно, даже несколько, так как сервисы получатся логически независимыми друг от друга, а для их быстрой совместной работы следует стоит выбрать правильный тип их взаимодействия.

Список литературы / References

[1]. High-load Systems. URL: https://flyoutsourcing.com/high-load-systems.html, accessed 17.06.2020.

[2]. Игумнов А.О., Сонькин Д.М. Об одном из подходов к оптимизации высоконагруженных систем на примере системы диспетчерского управления таксомоторным парком. Науковедение, 2013 г., №2(15), 7 стр. / Igumnov A.O., Sonkin D.M. One approache to optimize highload systems on example of dispatching taxi system. Naukovedenie, №2(15), 7 p. (in Russian).

[3]. Rajković P., Janković D., Milenković A. Using cqrs pattern for improving performances in medical information systems. In Proc. of the 6th Balkan Conference in Informatics, 2013, pp. 86-91.

[4]. Pacheco V. F. Microservice Patterns and Best Practices: Explore patterns like CQRS and event sourcing to create scalable, maintainable, and testable microservices. Packt Publishing, 2018, 366 p.

[5]. What are NoSQL databases? URL: https://aws.amazon.com/ru/nosql/, accessed 17.06.2020.

[6]. Обзор NOSQL баз данных / NOSQL Database Overview. URL: https://oracle-patches.com/db/3688-обзор-nosql-баз-данных, accessed 05.09.2020 (in Russian).

[7]. Redis. URL: https://redis.io/, accessed 28.10.2020.

[8]. ArangoDb, URL: https://www.arangodb.com/, accessed 28.10.2020.

[9]. What Is ClickHouse? URL: https://clickhouse.tech/docs/en/, accessed 28.10.2020.

[10]. MongoDb. URL: https://www.mongodb.com/, accessed 17.07.2020.

[11]. Postgres Professional. URL: https://habr.com/ru/company/postgrespro/blog/458186/, accessed 06.09.2020 (in Russian).

[12]. Design a DDD-oriented microservice. URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice, accessed 06.09.2020.

[13]. Apache Kafka. URL: https://kafka.apache.org/, accessed 28.10.2020.

[14]. Pattern: Command Query Responsibility Segregation (CQRS). URL: https://microservices.io/patterns/data/cqrs.html, accessed 25.06.2020.

[15]. Meyer B. Object-Oriented Software Construction. Prentice Hall, 1994, 534 p.

[16]. МТС ПОИСК / MTS Search. URL: https://poisk.mts.ru/, accessed 25.06.2020 (in Russian).

Информация об авторе / Information about the author

Василий Андреевич РУДОМЕТКИН – аспирант в области вычислительной техники и информатики, эксперт в области разработки высоконагруженных систем, сервисов геолокации, банковских систем. Сфера научных интересов: разработка и тестирование высоконагруженных систем.

Vasilii Andreevich RUDOMETKIN – postgraduate student in the field of computer technology and informatics, an expert in the development of high-load systems, geolocation services, banking systems. Research interests: development and testing of high-load systems.

DOI: 10.15514/ISPRAS–2020–32(6)–7

Внутрипроцедурный анализ для поиска ошибок на основе символьного выполнения

[1] *А.Е. Бородин, ORCID: 0000-0003-3183-9821 <alexey.borodin@ispras.ru>*
[1,2] *И.А. Дудина., ORCID: 0000-0002-5359-184X <eupharina@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. В работе описывается внутрипроцедурный анализ отдельных функций, использующийся в инструменте статического поиска ошибок Svace. Отличительные особенности анализа: анализ по графу потока управления, символьное выполнение с объединением состояний анализа в точках слияния путей, анализ только части путей в функциях с циклами, одновременный запуск всех анализаторов, моделирование достижимых ячеек памяти, нумерация значений переменных.

Ключевые слова: статический анализ; символьное выполнение; svace; поиск ошибок

Для цитирования: Бородин А.Е., Дудина И.А. Внутрипроцедурный анализ для поиска ошибок на основе символьного выполнения. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 87-100. DOI: 10.15514/ISPRAS–2020–32(6)–7

Symbolic Execution Based Intra-Procedural Analysis for Search for Defects

[1] *A.E. Borodin, ORCID: 0000-0003-3183-9821 <alexey.borodin@ispras.ru>*
[1,2] *Dudina I.A., ORCID: 0000-0002-5359-184X <eupharina@ispras.ru>*

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

Abstract. Svace is a static analysis tool for bug detection in C/C++/Java source code. To analyze a program, Svace performs an intra-procedure analysis of individual functions, starting from the leaves of a call-graph and moving towards the roots, and uses summaries of previously analyzed procedures at call-cites. In this paper, we overview the approaches and techniques employed by Svace for the intra-procedural analysis. This phase is performed by an analyzer engine and an extensible set of detectors. The core engine employs a symbolic execution approach with state merging. It uses value numbering to reduce the set of symbolic expressions, maintains points-to relationship graph for memory modeling, and performs strong and weak updates of program values. Detectors are responsible for discovering and reporting bugs. They calculate different properties of program values using a variety of abstract domains. All detectors work simultaneously orchestrated by the engine. Svace analysis is unsound and employs a variety of heuristics to speed-up. We designed Svace to analyze big projects (several MLOCs) in just a few hours and report as many warnings as possible, while keeping a good quality of reports ≥ 65 of true positives). For example, Tizen 5.5 (20MLOC) analysis takes 8.6 hours and produces 18,920 warnings, more than 70% of which are true-positive.

Keywords: static analysis; symbolic execution; svace; search for defects

For citation: Borodin A.E., Dudina I.A. Symbolic Execution Based Intra-Procedural Analysis for Search for Defects. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 87-100 (in Russian). DOI: 10.15514/ISPRAS-2020-32(6)-7

1. Введение

Статический анализатор Svace осуществляет поиск ошибок в программах, написанных на языках C/C++/Java [1, 2]. Анализатор за время, сравнимое с временем компиляции, осуществляет поиск ошибок с небольшим уровнем ложных срабатываний. Для анализа не требуется специальная подготовка программ, и не накладывается никаких ограничений на используемые конструкции языка.

Для поиска ошибок используются разные подходы: как анализ на основе абстрактного синтаксического дерева, так и межпроцедурный анализ с моделированием значений переменных и ячеек памяти.

На вход анализатору Svace подаётся исходный код вместе с командой сборки. Svace перехватывает команды запуска компилятора и компоновки. Затем запускается модифицированный компилятор (Clang для C/C++ [3] и OpenJDK javac для Java [4]). Компилятор строит абстрактное синтаксическое дерево (АСД) и запускает детекторы для поиска ошибок на АСД, а также генерирует промежуточное представление программы (LLVM биткод [5] для C/C++ и байткод для Java). Промежуточное представление подаётся на вход основному анализатору SvEng[1]. Основной анализатор строит граф вызовов и запускает поочерёдный анализ каждой функции начиная с листьев графа. В данной работе описывается используемый при этом внутрипроцедурный анализ отдельных функций, который является базой для межпроцедурного анализа.

В разд. 2 описывается обобщённый анализ на основе символьного выполнения [6], который может использоваться различными анализаторами для поиска ошибок в исходном коде программ. Приводится общая схема анализа, даётся описание основных используемых абстракций: идентификатор значения, ссылка, граф указателей, атрибут, абстрактное состояние. В подразделе 2.7 дано описание расширения анализа для поддержки чувствительности к путям. В разд. 3 описывается реализация анализа, используемая в инструменте Svace для внутрипроцедурного анализа функций.

2. Обобщённый анализатор

Разработанный анализ предназначен в первую очередь для выполнения неконсервативного анализа. Неконсервативным будем называть анализ, который для выполнения каких-либо требований (скорость анализа, потребление памяти, простота реализации и др.) в некоторых случаях может выдавать некорректные результаты.

2.1 Символьное выполнение с объединением состояний анализа

Опишем анализ для поиска ошибок на основе символьного выполнения с объединением состояний анализа в точках слияния путей.

Рассмотрим граф потока управления для некоторой функции. С рёбрами графа потока управления будем ассоциировать абстрактные состояния, описывающие интересующие нас свойства функции. Символьное выполнение осуществляется в топологическом порядке. Шаг анализа для каждой вершины графа из абстрактного состояния на её входном ребре формирует абстрактное состояние для выходного ребра. Анализ точек слияния путей для ациклических подграфов проводится после того, как получены состояния на входных рёбрах.

[1] Сокращение от Svace Engine – движок анализатора Svace.

Анализ формирует абстрактное состояние на выходном ребре функции, которое будет описывать свойства программы для всех рассматриваемых путей.

Для анализа сильно связанных компонент (ССК) графа потока управления производится несколько итераций цикла с сохранением всех абстрактных состояний на выходных рёбрах из ССК. После анализа ССК, анализ выполняет объединение состояний на выходных рёбрах. При этом используются эвристики для получения состояния, которое будет описывать все возможные пути выполнения ССК. В этом случае анализ может некорректно описывать свойства для части путей, если эвристики сработали неверно.

Если внутри ССК можно выделить внутреннюю ССК, то на каждом шаге анализа внешней ССК производится несколько итераций анализа внутренней ССК. Такая схема анализа циклов имеет экспоненциальную сложность от количества вложенных циклов. Чтобы уменьшить сложность для внутренних ССК уменьшается количество обходов. При достижении некоторого порога вложенности анализатор перестаёт выделять внутренние ССК.

Будем использовать следующие обозначения:

- S – множество символов языка,
- I – множество инструкций,
- P – множество вершин графа потока управления,
- E – рёбра графа потока управления,
- Γ – абстрактные состояния.

Анализ параметризуется следующими компонентами:

- дополнительные анализы A_i, на основе которых выдаются предупреждения;
- детекторы C_i для поиска ошибок;
- передаточные функции для каждой инструкции $T[I]$;
- количество обходов ССК N;
- функции для создания состояния для точек слияния путей $U[P]$: $\Gamma \times \Gamma \longmapsto \Gamma$;
- функции для создания состояния из нескольких состояний для выходных рёбер ССК $U'[P] : \Gamma \times \Gamma \longmapsto \Gamma$.

Для каждой инструкции выполнение всех дополнительных анализов и детекторов производится одновременно.

Детектор запускается для каждой посещённой вершины и на основе абстрактного состояния на входном ребре выдаёт предупреждение об ошибке. Дополнительные анализы формируют абстрактные состояния на выходном ребре.

Любому реализованному анализу и детектору при проходе через некоторую вершину доступны результаты всех других анализов на входных рёбрах этой вершины, но недоступны результаты на выходных рёбрах. Важно, что последовательность применения анализов не должна влиять на результат.

Подобная реализация имеет следующие преимущества.

- Высокая скорость работы. Общие действия выполняются каким-либо одним анализатором.
- Возможность быстрого написания довольно сложных детекторов за счёт того, что каждому детектору доступны все проанализированные свойства.

2.2 Идентификаторы значений

Идентификатор значения[2] является абстракцией для обозначения разбиения значений переменных на классы эквивалентности для шага символьного выполнения, и подчиняется следующим правилам.

I. Если двум переменным сопоставлен один и тот же идентификатор значения, то и при выполнении эти переменные будут иметь одинаковые значения (задача нумерации значений[3]).

II. Для инструкции идентификаторы значения на входном и выходном ребрах равны для всех переменных и ячеек памяти, кроме тех, значения которых меняются в данной инструкции. Данное требование к свойствам идентификаторов значений позволяет значительно упростить написание передаточных функций: все созданные свойства не теряют корректность по мере выполнения анализа.

Идентификаторы значений играют центральную роль в описываемом анализе. Помимо задачи нумерации значений, идентификаторы значений используются для описания большинства анализируемых свойств значений переменных. При этом свойства ассоциируются с идентификаторами значений. Сами свойства в свою очередь могут описываться с помощью идентификаторов значений.

Для описания свойств используются *атрибуты*. Атрибуты могут ассоциироваться с идентификаторами значений и с рёбрами графа потока управления для некоторого абстрактного состояния. Атрибут обозначает некоторое анализируемое свойство (интервал значений переменной, необходимое условие достижимости точки, список идентификаторов значений заблокированых мьютексов и др). Атрибуты должны иметь функцию объединения двух атрибутов $\sqcup$. Эта функция используется при объединении состояний в точках слияния путей. Во многих случаях удобно, чтобы атрибуты были решётками, или полурешётками, а функция объединения – наименьшим верхним элементом. Но это не является требованием для атрибута.

Атрибуты позволяют разделять абстрактное состояние между дополнительными анализами – каждый такой анализ работает со своим множеством атрибутов, поэтому не возникает конфликтов, когда несколько анализов по-разному формируют выходное состояние.

Абстрактное состояние содержит информацию о значении всех атрибутов для идентфикаторов значений. Для наглядности рассмотрим атрибуты: интервал значений **VI**, который описывает интервал возможных значений переменной, и атрибут **Null**, обозначающий что переменная имеет нулевое значение. Тип атрибута будем выделять жирным шрифтом, а значение интервала курсивным. VI – тип атрибута, а **VI** – множество значений.

Обозначим V – множество идентификаторов значений. Функция $val\colon S \hookrightarrow V$ для каждого символа возвращает ассоциированный идентификатор значения. Функция val является частью абстрактного состояния. Также в абстрактном состоянии для каждого типа атрибута и идентификатора значений содержится информация о значении атрибута:

– $\Gamma[\mathbf{VI}]\colon V \mapsto VI$,
– $\Gamma[\mathbf{Null}]\colon V \mapsto Null$.

Вместо записи $\Gamma(\mathbf{Null}, V)$ будем использовать сокращения $\Gamma[\mathbf{Null}](V)$ или $\mathbf{Null}(V)$.

[2] Идентификатор значения является символьной переменной. В данной статье будем использовать этот термин для обозначения символьных переменных вместе со способом их использования в анализаторе.

[3] Задача нумерации значений заключается в определении множества переменных, значения которых равны в данной точке для всех путей выполнения. Для случая символьного выполнения все пути выполнения можно заменить на все рассматриваемые пути выполнения.

Рассмотрим фрагмент кода на языке C, иллюстрирующий преимущества ассоциирования атрибутов не с переменными программы, а с идентификаторами значений (Листинг 1).

```
1:     void func(int*p) {
2:           int*q = p;
3:           if(!q) {
4:                 *p = 7;
5:           }
6:     }
```

Листинг 1. Разыменование нулевого указателя
Listing 1. Null pointer dereference

После анализа инструкции на строке 2 в абстрактном состоянии выполняется $val(q) = val(p)$. Пусть $val(q) = v_1$. При анализе инструкции на строке 3 идентификатору значения переменной q будет сопоставлено значение атрибута **Null**, обозначающее, что указатель имеет только нулевое значение $null$. При анализе инструкции разыменования на строке 4 абстрактное состояние будет иметь следующие свойства:

$$val(p) = v_1, val(q) = v_1,\ \Gamma[\mathbf{Null}](v_1) = null\,.$$

Фактически, анализу известно, что разыменовывается указатель, значение которого может быть только нулём. Этого достаточно, чтобы выдать предупреждение о разыменовании нулевого указателя. Если строка 4 достижима, то произойдёт ошибка.

Поскольку идентификаторы значений обозначают значения, которые не меняются в разных точках программы, многие свойства удобно выражать, используя идентификаторы значений. Иначе говоря, значения атрибутов ссылаются на идентификаторы значений. Идентификаторы значения при этом используются как символы некоторого алфавита.

Рассмотрим атрибут pt (см. подразд. 2.4), значениями которого являются множество идентификаторов значений для адресов указываемых ячеек.

```
1:     void func(int f) {
2:           int a, b, c;
3:           int*p = f>0? &a : f<0? &b : &c;
4:           int*q = p;
5:           p = 0;
6:     }
```

Листинг 2. Пример с присваиванием
Listing 2. Example with assignment

Состояние в точке 3 на Листинге 2 будет имеет следующие свойства:

$$val(\&a) = v_a, val(\&b) = v_b, val(\&c) = v_c,$$
$$val(p) = v_{pp}, pt(v_{pp}) = \{v_a, v_b, v_c\}.$$

Т.е. значение переменной p указывает на адреса переменных a, b и c.

Эти же свойства будут выполняться и для строки 6, несмотря на то, что значение переменной p поменялось:

$$val(p) = v_0, val(q) = v_{pp}, pt(v_{pp}) = \{v_a, v_b, v_c\}, \Gamma[\mathbf{Null}](v_0) = null.$$

Таким образом, анализу достаточно поменять информацию о значении переменной p, но не требуются менять свойства значений этой переменной, что позволяет оптимизировать время работы анализа.

2.3 Ссылки

Ссылками будем называть подмножество идентификаторов значений, для которых выполняется два условия:

I. значение является указателем;
II. анализ выполняет моделирование памяти для этого указателя, т.е. отслеживаются значения, находящиеся в указываемой ячейке памяти.

Расширим функцию val для моделирования памяти: $val: S \cup R \hookrightarrow V$, где R – множество ссылок. Запись $val(s) \mapsto v_1$, $val(v_1) \mapsto v_2$ означает, что переменная s имеет значение, описываемое идентификатором v_1, а значение в ячейке, на которую указывает s описывается идентификатором v_2.

Ссылки используются для моделирования указателей и всегда обозначают указатели. Моделируются только те указатели, для которых анализ может отследить, что они не являются алиасами. В наиболее простой реализации ссылки обозначают адреса локальных переменных, про которые точно известно, что они не равны друг другу.

2.4 Граф указателей

Анализ указателей позволяет ответить на вопрос, на какие моделируемые ячейки памяти указывают переменные. Результат этого анализа удобно представлять в виде графа указателей.

Графом указателей назовём направленный граф $P = < V, pt >$, вершинами которого являются идентификаторы значений, а рёбра идут от идентификаторов значений к ссылкам. Рёбра задаются функцией $pt: V \hookrightarrow 2^R$, которая для идентификатора значений возвращает множество указываемых ссылок. Если граф указателей имеет ребро $< v_1, r_2 >$, то это означает, что значение, обозначаемое идентификатором v_1, указывает на ячейку памяти, моделируемую ссылкой r_2. Либо, что тоже самое, значение v_1 может иметь алиас r_2.

Функция pt является ещё одной частью абстрактного состояния. Анализ указателей необходим для моделирования непрямых обращений к памяти. К анализу указателей не предъявляется каких-либо особых требований и его можно рассматривать как ещё одну параметризацию анализа.

2.5 Сильные и слабые обновления

Если анализ инструкции присваивания значения переменной или ячейке памяти может уничтожить эффект от присваивания всех предыдущих значений, то такое поведение называют сильным обновлением. Если же эффекты предыдущущих присваиваний всё ещё учитываются анализом, то происходит слабое обновление.

В описываемом анализе сильные обновления можно производить для всех переменных и ячеек, для которых в графе указателей есть только одно исходящее ребро.

Рассмотрим инструкции языка С, выполняющие непрямые обращения к памяти: $r = *p$, $*p = a$. Опишем передаточные функции для этих инструкций для произвольного анализа с сохранением результатов в атрибут $\mathbf{A}_i$.

Инструкция чтения значения из памяти: $m_n = *p$. Пусть $pt(val(p)) = Pt$. Возможны 3 случая: множество Pt имеет один элемент, множество имеет более одного элемента и множество пусто. В первом случае пусть $Pt = \{m\}$, можно сделать сильное обновление, в результате в выходном состоянии $val(r) = val(m)$.

Для случая множества элементов $Pt = \{m_1, m_2, \ldots, m_n\}$ создадим новый идентификатор значения v_r, который сопоставим r в выходном состоянии $val(r) = val(v_r)$. Для всех атрибутов будем использовать функцию объединения свойств:

$$\Gamma[\mathbf{A}_i](v_r) = \sqcup(\Gamma[\mathbf{A}_i]val(m_1), \ldots, \Gamma[\mathbf{A}_i]val(m_n)).$$

Если множество пусто, то с переменной r будет ассоциирован новый идентификатор значения.

Инструкция записи в память: $*p = a$. Аналогично рассмотрим 3 случая в зависимости от множества Pt. Если $Pt = \{m\}$, то в выходном состоянии $val(m) = val(a)$. Для случая множества элементов $Pt = \{m_1, m_2, \ldots, m_n\}$ выполним слабое обновление, для этого для каждой ячейки памяти создадим новый идентификатор значения v_i, который сопоставим

ячейкам в выходном состоянии $val(m_i) = v_i$. Для всех атрибутов будем использовать функцию объединения свойств:

$$\Gamma[\mathbf{A}_i](v_r) = \Gamma[\mathbf{A}_i](val(a) \sqcup \Gamma[\mathbf{A}_i](prev(m_i))).$$

Здесь функция $prev$ вернёт значение $val(m_i)$, если оно определено, либо создаст новый идентификатор значения. Т.е. для каждой из возможных ссылок учитывается, что её значение могло измениться. Если множество пусто, то с переменной $val(p)$ будет ассоциирован новый идентификатор значения.

Для создания абстрактного состояния для точки слияния путей используется похожее правило, как для инструкции чтения значения из памяти. Правило для значений моделируемых ячеек памяти объединяет свойства из входных состояний.

$$\Gamma[\mathbf{A}_i](joinval(m)) = \Gamma_1[\mathbf{A}_i](val(m) \sqcup \Gamma_2[\mathbf{A}_i](prev(m_i))).$$

Функция $joinval$ возвращает идентификатор значения ссылки m, если в обоих состояниях для входных рёбер ссылке m присвоен один и тот же идентификатор значения; и создаёт новый идентификатор значения в остальных случаях.

2.6 Ядро анализа и дополнительные анализы

Все дополнительные анализы реализуются в виде обработчиков инструкций, оперирующих не переменными, а соответствующими идентификаторами значений. Ядро анализа отслеживает граф указателей, выполняет сильные, либо слабые обновления, и вызывает обработчики для соответствующих ситуаций. Фактически ядро анализа занимается следующими компонентами абстрактных состояний: val, pt, т.е. отслеживает значения переменных и осущесьвляет анализ указателей.

Рассмотрим анализ на небольшом примере (Листинг 3), где в качестве дополнительных анализов будем использовать анализ интервалов.

```
1:     int f(int a, int*p) {
2:           int x = 1;
3:           *p = x;
4:           if(a)
5:                 *p = 2;
6:           return *p;}
```

Листинг 3. Небольшой пример
Listing 3. Small Example

Абстрактные состояния анализа после соответвующих строк:

2: $val(x) = v_x, \mathbf{VI}(v_x) = [1; 1]$.

3: $\Gamma_2, val(p) = v_p, val(v_p) = v_x$.

5: $\Gamma_4, val(v_p) = v_2, \mathbf{VI}(v_2) = [1; 2]$.

6: $\Gamma_5, val(v_p) = v_{p2}, val(ret) = v_{p2}, \mathbf{VI}(v_{p2}) = [1; 2]$.

Значение атрибута **VI** в последнем абстрактном состоянии было получено применением функции объединения атрибутов для значений по правилу, описанному в подразд. 2.5 для точек слияния:

$$\Gamma[\mathbf{VI}](joinval(v_p)) = \Gamma_1[\mathbf{VI}](val(v_p) \sqcup \Gamma_2[\mathbf{VI}](prev(v_p))),$$

$$\Gamma[\mathbf{A}_i](joinval(v_{p2})) = \Gamma_1[\mathbf{A}_i](val(v_x) \sqcup \Gamma_2[\mathbf{A}_i](prev(v_2))) = [1; 1] \sqcup [2; 2] = [1; 2].$$

2.7 Чувствительность к путям

Анализ будем называть чувствительным к путям, если он способен различать пути выполнения программы в процессе анализа. Описываемая реализация чувствительности к путям основана на расширении использования идентификаторов значений, которые

используются как кирпичики в условных выражениях, и в определениях идентификаторов значений. Задача чувствительности к путям – отсеять пути, которые не выполнимы из-за несовместных условий.

Для конкретного анализатора добавление чувствительности к путям позволяет не только уменьшить количество ложных предупреждений, связанный с несовместными условиями, но также увеличить количество выдаваемых истинных предупреждений. Последнее достигается за счёт выдачи предупреждений в сложных случаях, для которых анализатор без чувствительности к путям не может обеспечить приемлемое качество анализа.

Например, в программе из Листинга 4 путь, проходящий через два разыменования p, не является выполнимым.

```
1:     void f(int a, int**p) {
2:             int x = a+2;
4:             if(a>10) {
5:                     *p = 0;
6:             }
7:             if(x<12) {
8:                     **p = 0;
9:             }
10:    }
```

Листинг 4. Функция с невыполнимым путём
Listing 4. Function with infeasible path

Если бы путь являлся выполнимым, то на строке 8 произошло бы разыменования нулевого указателя $* p$.

Условным будем называть атрибут, который характеризует условия наступления некоторых событий. Значением атрибута является формула – выражение логики высказываний, где могут использоваться константы языка программирования и идентификаторы значений для обозначения свойств значений переменных. Пример условия: $(v_x > v_y) \wedge (v_x \neq 0) \vee (v_y < 10)$.

Имеется предопределённый условный атрибут **Ness** – необходимые условия достижимости ребра графа потока управления.

Остальные атрибуты используются отдельными детекторами по следующей схеме. В точке, где происходит интересующее событие, значение атрибута C_i устанавливается в $true$ и ассоциируется с некоторым идентификатором значения. В точках слияния путей 1 и 2 значение атрибута вычисляется по следующей формуле:

$$\Gamma_{res}[C_i](v) = (\Gamma_1[\mathbf{Ness}](v) \wedge \Gamma_1[C_i](v)) \vee (\Gamma_2[\mathbf{Ness}](v) \wedge \Gamma_2[C_i](v)).$$

Для отсеивания несуществующих путей – перед выдачей предупреждения об ошибке – запускается SMT-решатель, которому на вход подаётся конъюнкция необходимого условия с условием отслеживаемого атрибута. Если SMT-решатель возвращает unsat, то ошибка на таких путях не существует.

Для примера из Листинга 4 значение атрибута **Ness** будет $v_x = v_a + 2 \wedge v_x < 12$, значение атрибута, отслеживающего присваивание нулевого указателя, – $v_a > 10$. Поскольку формула $v_x = v_a + 2 \wedge v_x < 12 \wedge v_a > 10$ не имеет модели, благодаря SMT-решателю, не будет выдано ложное срабатывание о разыменовании нуля.

Заметим, что передаточные функции для анализируемых свойств и необходимых условий, а также сам вид условных атрибутов, зависят от конкретной реализации. Более простая реализация может не добавлять условие $v_x = v_a + 2 \wedge v_x < 12$ в необходимые условия. В этом случае путь не будет отсеян. Фактически, это будет чувствительный к путям анализ, без чувствительности по данным.

SMT-решатель вызывается только тогда, когда имеется подозрение на ошибку. Если формула не разрешима, то предупреждение не будет выдано, во всех остальных случаях предупреждение будет выдано. SMT-решатель не вызывается, чтобы подсчитать какие-либо промежуточные данные.

Описанная реализации, чувствительного к путям анализа, имеет следующие преимущества:

- простота расширения для анализа на основе идентификаторов значений;
- высокая скорость, поскольку SMT-решатель вызывается только в том случае, когда есть подозрение на ошибку;
- отсутствие ограничений на конкретный вид анализируемых формул.

3. Реализация анализа в Svace

3.1 Используемые эвристики

Анализ не является консервативным и может использовать эвристики как для повышения точности анализа (иногда в ущерб корректности), так и для уменьшения времени анализа. Основные используемые эвристики:

- входные параметры процедуры, а также их смещения и разыменования, не являются алиасами;
- выбранное множество путей анализа описывает все существенные пути анализа (путь считаем существенным, если он может повлиять на результат анализа).

Анализ покрывает все возможные пути в функциях без циклов, и только часть путей в функциях с циклами.

Дополнительно используется ряд ограничений на различные параметры:

- тело цикла обходится только 2 раза; большее количество обходов не используется, чтобы не замедлять анализ;
- максимальное моделируемое количество ссылок для одного идентификатора значений 150;
- максимальное количество моделируемых неконстантных смещений для одного указателя 10;
- максимальная длина цепочки моделируемых разыменований и смещений для переменных равна 6;
- максимальное время анализа одной функции 5 минут.

Ограничение на время анализа одной функции является защитой от нестандартного кода. На всех известных нам проектах анализ всех функций укладывается в 5 минут.

3.2 Анализируемый язык svace0

Анализ выполняется для внутреннего представления на языке svace0. Язык svace0 представляет собой упрощённую версию языка LLVM с дополнительными инструкциями, позволяющими получить больше информации о программе.

Для анализа программ, написанных на языках C/C++, производится трансляция в промежуточное представление LLVM, которое затем преобразуется в язык svace0.

Для анализа программ, написанных на языке Java, производится трансляция в байткод, который затем преобразуется в язык svace0. Реализация большей засти детекторов и анализов не отличается для C/C++ и Java.

Язык является довольно низкоуровневым, что с одной стороны позволяет точно моделировать семантику программ, а с другой стороны затрудняет анализ некоторых высокоуровневых конструкций.

3.3 Используемые атрибуты

Одновременный запуск всех анализов и возможность использования результатов других анализов позволяют получать выигрыш как по времени выполнения анализа, так и по используемой памяти. Высокая скорость анализа обеспечивается за счёт того, что нет необходимости выполнять похожие вычисления, можно сразу использовать результаты других анализов. Минимизация использования памяти достигается за счёт того, что анализу не требуется хранить большинство абстрактных состояний: как правило, после анализа инструкции состояние на входном ребре больше не потребуется и может быть удалено.

Результаты анализов сохраняются в виде атрибутов. Добавление нового атрибута не требует значительных затрат вычислительных ресурсов. В настоящий момент в Svace используется более 350 атрибутов. Приведём некоторые виды атрибутов в качестве примера:

- возможный интервал значений целочисленный переменных;
- интервал размера массива и интервал смещения указателя на массив;
- мьютекс был заблокирован;
- переменная получена из непроверенного источника;
- интервал длины строки;
- указатель на динамическую память сравнили с константой;
- условия, при которых была выделена динамическая память;
- условия, при которых переменная не была инициализирована;
- условия того, что указателю присвоено нулевое значение (требуется для поиска разыменования нулевых указателей);
- условия того, что переменная может иметь нулевое значение (требуется для поиска ошибок деления на ноль);
- необходимые условия достижения точки в программе.

3.4 Другие анализы

В анализаторе svace описанный анализ является только небольшой частью, которая требуется для внутрипроцедурного анализа. Внутрипроцедурный анализ является базой для межпроцедурного анализа на основе резюме. На основе межпроцедурного анализа выполнен анализ конструкторов, деструкторов и операторов присваивания классов C++ для обнаружения неконсистентности их реализации [7].

Кроме этого перед запуском внутрипроцедурного анализа для каждой функции выполняется анализ потока данных, консервативно вычисляющий важные свойства функции (недостижимый код, функции, завершающие программу, живые переменные) [8].

Дополнительно используются анализы на основе абстрактного синтаксического дерева для реализации части детекторов. Эти анализы запускаются в модифицированных компиляторах (clang, javac).

3.5 Результаты

В табл. 1 приведено время анализа для проектов с открытым исходным кодом. Измерялось только время анализатора SvEng. Во время анализа были включены все реализованные детекторы.

Анализ производился на двух серверах, имеющих следующие характеристики:

- Сервер 1: Intel Xeon CPU E5-2650 2.00GHz, 32 ядра, 256 Гб ОП;
- Сервер 2: Intel Core CPU i7-6700 3.40GHz, 8 ядер, 32 Гб ОП.

Операционные системы tizen и android имеют существенный размер исходного кода, их анализ желательно проводить на сервере, имеющем хотя бы 32 Гб оперативной памяти.

Текущая скорость анализа позволяет проанализировать эти проекты во время ночной сборки. Для сравнения в таблице приведено время сборки этих проектов на сервере 1.

Табл. 1. Analysis time for big projects
Table 1. Время анализа больших проектов

Проект	**Размер кода, тыс. строк**	**Время анализа, мин.**		**Время сборки, мин.**
		сервер 1	**сервер 2**	
tizen 5.5	19 988	272	516	250
android 5	8 561	236	421	31
android 9 java	12 122	27	32	77

Для значительной части небольших проектов для анализа требуется не больше 2Гб оперативной памяти. В табл. 2 приведены данные анализа проектов busybox, cairo, xorg-server и nss, имеющих размер от 139 до 393 тысяч строк кода, на сервере 1 с ограничением используемой памяти.

Табл. 2. Analysis time for small projects
Table 2. Время анализа небольших проектов, сек.

Память, Гб	**Проект**			
	busybox-1.18.5	**cairo-1.12.14**	**xorg-server-1.12.3**	**nss-3.17.4**
	Размер, тыс. строк кода			
	139	270	393	355
16	310	176	524	336
8	314	160	521	336
4	319	156	520	335
2	338	158	594	378
1	419	231	1430	418

Размер исходного кода в строках позволяет примерно оценить сложность проекта. Скорость анализа зависит от сложности кода и используемых конструкций. Как видно из результатов скорость анализа варьируется от 448 до 1534 строк в секунду при использовании 16Гб оперативной памяти. Снижение доступной памяти до 2Гб практически не влияет на время анализа. Худший результат был 274 строк в секунду для проекта xorg-server при ограничении доступной памяти в 1 Гб, при этом увеличение доступной памяти до 2 Гб ускорило анализ этого проекта в 2,4 раза.

Табл. 3. Качество выдаваемых предупреждений
Table 3. Analysis quality

Проект	Всего	Размечено	Истинных
cairo-1.12.14	321	10.9%	94.2%
xorg-server-1.12.3	791	10.1%	78.7%
nss-3.17.4	826	22%	85.1%
busybox-1.18.5	1561	10.1%	80.5%
android 9 java	7327	10.68%	77.7%
android 5.02	10 414	10.7%	76.4%
tizen 5.5	18 920	22.1%	70.8%

Важной характеристикой анализатора является процент истинных предупреждений. Фактически постановка задачи – находить настолько много предупреждений, насколько можно при обеспечении приемлемого уровня истинных срабатываний. В табл. 3 представлены данные по качеству предупреждений среди 190 стабильных детекторов. В таблицу входят данные только для анализатора SvEng. В колонке «Всего» приводится общее

количество предупреждений, выданных для оцениваемых детекторов, колонка «Размечено» содержит процент размеченных вручную предупреждений среди выданных, и колонка «Истинных» показывает, сколько процентов предупреждений среди всех размеченных было классифицировано как истинные.

4. Похожие работы

Инструменты, основанные на символьном выполнении без объединения состояний в точках слияния путей: PREfix [9], Archer [10] и, по всей видимости, Prevent [11]. Наиболее ранней работой является описание ин-струмента PREfix, в котором по умолчанию анализируется 50 путей. Количество выдава-емых предупреждений почти перестаёт изменяться после рассмотрения больше 100 путей внутри функции.

Использование символьного выполнения без объединения путей имеет как свои достоинства, так и недостатки. Очевидным недостатком является проблема экспоненциального роста путей внутри функции. Из-за чего инструмент не может проанализировать значительную часть путей в функциях с большим количеством путей. Кроме этого общее время анализа выше по сравнению со схемой с объединением путей. Другой проблемой является создание резюме для анализируемой функции. Либо придётся писать дополнительный анализ для создания резюме, либо ограничиться резюме, описывающими неполное поведение функции (так как не все пути проанализированы). К преимуществам можно отнести то, что каждый путь моделируется более точно, нет необходимости в эвристической функции объединения абстрактных состояний, реализация многих детекторов упрощается. Помимо этого, легче сообщить об ошибке пользователю, поскольку достаточно показать рассматриваемый путь.

Инструмент Saturn [12] выполняет объединение состояний в точках слияния путей. Используется SAT-решатель. Детекторы запускаются по очереди. Способ анализа циклов зависит в том числе и от детектора. Абстрактные состояния между детекторами не разделяются. Время анализа существенно зависит от включённых детекторов. Похожую архитектуру имеет инструмент Calysto [13], основанный на SMT-решателе, имеющий лучшую скорость и точность.

SharpChecker [14, 15] – разрабатываемый в ИСП РАН инструмент для анализа программ на языке C\#. Инструмент имеет множество общих черт с описанной схемой (объединение состояний анализа в точках слияния путей, одновременный запуск всех анализаторов, моделирование достижимых ячеек памяти), при этом он изначально проектировался из расчёта использования вместе с SMT-решателем. Текущая схема анализа циклов, используемая в Svace, взята из этой работы. В более ранних версиях Svace объединение состояний производилось на обратных рёбрах.

5. Заключение

Описан внутрипроцедурный анализ, позволяющий относительно быстро и качественно анализировать большие объёмы исходного кода, что подтверждается реализацией в инструменте Svace.

Описана общая схема анализа, которая может быть улучшена и дополнена множеством способов:

- добавление девиртуализации для построения более точного графа вызовов;
- улучшение анализа указателей, в том числе использование анализа указателей с чувствительностью к путям;
- более точное моделирование циклов;
- уменьшение количества моделируемых ячеек памяти и значений переменных для ускорения анализа.

Список литературы / References

[1] В.П. Иванников, А.А. Белеванцев, А.Е. Бородин, В.Н. Игнатьев, Д.М. Журихин, А.И. Аветисян, М.И. Леонов. Статический анализатор Svace для поиска дефектов в исходном коде программ. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 231-250 / V.P. Ivannikov, A.A. Belevantsev, A.E. Borodin, V.N. Ignatyev, D.M. Zhurikhin, A.I. Avetisyan, M.I. Leonov. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 231-250 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-7.

[2] A. Borodin, A. Belevantsev, D. Zhurikhin, and A. Izbyshev. Deterministic static analysis. In Proc. of the 2018 Ivannikov Memorial Workshop, 2018, pp. 10-14. DOI: 10.1109/IVMEM.2018.00009.

[3] Clang. URL: https://clang.llvm.org, accessed September 10, 2020.

[4] The javac compiler. URL: https: //docs.oracle.com/en/java/javase/11/tools/javac.html, accessed September 10, 2020.

[5] LLVM bitcode. URL: https: //releases.llvm.org/8.0.1/docs/BitCodeFormat.html, accessed September 10, 2020.

[6] J. C. King. Symbolic execution and program testing. Communications of the ACM, vol. 19, no. 7, 1976, pp. 385-394.

[7] А.Е. Бородин и А.А. Белеванцев. Статический анализатор Svace как коллекция анализаторов разных уровней сложности. Труды ИСП РАН, том 27, вып. 2, 2015 г., стр. 111-134 / A.E. Borodin, A.A. Belevancev. A Static Analysis Tool Svace as a Collection of Analyzers with Various Complexity Levels. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 6, 2015, pp.111-134 (in Russian). DOI: 10.15514/ISPRAS-2015-27(6)-8.

[8] Р.Р. Мулюков, А.Е. Бородин. Использование анализа недостижимого кода в статическом анализаторе для поиска ошибок в исходном коде программ. Труды ИСП РАН, том 28, вып. 5, 2016 г., стр. 145-158 / R.R. Mulyukov, A.E. Borodin. Using unreachable code analysis in static analysis tool for finding defects in source code. Trudy ISP RAN/Proc. ISP RAS, 2016, vol. 28, issue 5, 2016, pp. 145-158 (in Russian). DOI: 10.15514/ISPRAS-2016-28(5)-9.

[9] W.R. Bush, J.D. Pincus, and D.J. Sielaff. A static analyzer for finding dynamic programming errors. Software-Practice and Experience, vol. 30, no. 7, 2000, pp. 775-802.

[10] Y. Xie, A. Chou, and D. Engler. Archer: using symbolic, path-sensitive analysis to detect memory access errors. ACM SIGSOFT Software Engineering Notes, vol. 28, no. 5, 2003, pp. 327–336.

[11] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler. A few billion lines of code later: using static analysis to find bugs in the real world. Communications of the ACM, vol. 53, no. 2, 2010, pp. 66–75.

[12] A. Aiken, S. Bugrara, I. Dillig, T. Dillig, B. Hackett, and P. Hawkins. An overview of the saturn project. In Proc of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, 2007, pp. 43–48. ACM.

[13] D. Babic and A.J. Hu. Calysto: scalable and precise extended static checking. In Proc. of the 30th International Conference on Software Engineering, 2008, pp. 211–220.

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

[15] V. Koshelev, V. Ignatiev, A. Borzilov, and A. Belevantsev. Sharpchecker: static analysis tool for C# programs. Programming and Computer Software, vol. 43, no. 4, 2017, pp. 268–276. DOI: 10.1134/S0361768817040041.

Информация об авторах / Information about authors

Алексей Евгеньевич БОРОДИН – кандидат физико-математических наук, научный сотрудник. Сфера научных интересов: статический анализ исходного кода программ для поиска ошибок.

Alexey Evgenevich BORODIN – PhD, researcher. Research interests: static analysis for finding errors in source code.

Ирина Александровна ДУДИНА – кандидат физико-математических наук, сотрудник ИСП РАН и ассистент кафедры системного программирования факультета ВМК МГУ. Сфера научных интересов: статический анализ, символьное исполнение, SMT-решатели.

Irina Aleksandrovna DUDINA – PhD, employee of ISP RAS and assistant of the Department of System Programming of the Faculty of CMC, Moscow State University. Research interests: static analysis, symbolic execution, SMT solvers.

DOI: 10.15514/ISPRAS–2020–32(6)–8

Практическая абстрактная интерпретация бинарного кода

[1,2] *М.А. Соловьев, ORCID: 0000-0002-0530-6442 <icee@ispras.ru>*
[1] *М.Г. Бакулин, ORCID: 0000-0002-8569-7382 <bakulinm@ispras.ru>*
[2] *С.С. Макаров, ORCID: 0000-0003-0077-237X <smakarov@ispras.ru>*
[2] *Д.В. Манушин, ORCID: 0000-0001-8985-4114 <dman95@ispras.ru>*
[1,2] *В.А. Падарян, ORCID: 0000-0001-7962-9677 <vartan@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. Математический аппарат абстрактной интерпретации предоставляет универсальный способ формализации и изучения алгоритмов анализа программ для самого широкого спектра прикладных задач. Однако его применение для практически значимых задач анализа бинарного кода связано с большим числом вызовов, как научных, так и инженерных. В настоящей работе предложены подходы к преодолению части этих трудностей. Описано промежуточное представление, учитывающее особенности бинарного кода. Предложена инфраструктура абстрактной интерпретации с возможностью выполнения интерпретации как вдоль отдельного пути в программе, так и статически до достижения неподвижной точки. В совокупности промежуточное представление и инфраструктура абстрактной интерпретации позволяют задать модель конвейера процессора, что позволяет проводить анализ бинарного кода для различных архитектур. В работе также представлены предварительные эксперименты и указаны дальнейшие направления развития проекта.

Ключевые слова: абстрактная интерпретация; анализ бинарного кода; динамический анализ; символьное выполнение; статический анализ

Для цитирования: Соловьев М.А., Бакулин М.Г., Макаров С.С., Манушин Д.В., Падарян В.А. Практическая абстрактная интерпретация бинарного кода. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 101-110. DOI: 10.15514/ISPRAS–2020–32(6)–8

Благодарности: работа поддержана грантом РФФИ № 18-07-01256.

Practical Abstract Interpretation of Binary Code

[1,2] *M.A. Solovev, ORCID: 0000-0002-0530-6442 <icee @ispras.ru>*
[1] *M.G. Bakulin, ORCID: 0000-0002-8569-7382 <bakulinm @ispras.ru>*
[2] *S.S. Makarov, ORCID: 0000-0003-0077-237X <smakarov @ispras.ru>*
[2] *D.V. Manushin, ORCID: 0000-0001-8985-4114 <dman95 @ispras.ru>*
[1,2] *V.A. Padaryan, ORCID: 0000-0001-7962-9677 <vartan @ispras.ru>*

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

Abstract. The mathematical foundations of abstract interpretation provide a unified method of formalization and research of program analysis algorithms for a broad spectrum of practical problems. However, its practical usage for binary code analysis faces several challenges, of both scientific and engineering nature. In this paper we address some of those challenges. We describe an intermediate representation that is tailored to binary code analysis; unlike some other IRs it is still useable in system code analysis. To achieve this, we take into account the low-level specifics of how CPUs work; on the IR level this mostly pertains to modeling main memory in that accesses can fail, and addresses can alias. Further, we propose an infrastructure for carrying out abstract interpretation on top of the IR. The user needs to implement the abstract state and the transfer functions, and the infrastructure handles the rest: two executors are currently implemented, one for analysis of a single path, and one for fixed point analysis. Both executors handle interprocedural analysis internally, via inlining or using summaries, so the interpretations only consider only procedure at a time, which greatly simplifies implementation. The IR and the abstract interpretation framework are used together to define a model pipeline for a target instruction set architecture, consisting of a fetch stage, a decode stage, and an execute stage. A distinct fetch stage allows to model delay slots, hardware loops, etc. We currently have limited implementations for RISC-V and x86. The x86 implementation is evaluated in two experiments where concolic execution is used to automatically analyze a «crackme» program, both in dynamic (execution trace) and static (executable image) setting. In conclusion, we outline the future directions of our project.

Keywords: abstract interpretation; binary code analysis; dynamic analysis; static analysis; symbolic execution

For citation: Solovev M.A., Bakulin M.G., Makarov S.S., Manushin D.V., Padaryan V.A. Practical abstract interpretation of binary code. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 101-110 (in Russian). DOI: 10.15514/ISPRAS-2020-32(6)-8

Acknowledgements: this work was supported by RFBR grant no. 18-07-01256.

1. Введение

Предложенный Патриком и Радией Кузо (Patrick Cousot, Radhia Cousot) аппарат абстрактной интерпретации [1] позволяет проводить анализ программ с целью исследования определенных свойств. Основная идея абстрактной интерпретации заключается в аппроксимации возможных состояний в точках программы с использованием абстрактного домена, где каждое абстрактное состояние соответствует множеству конкретных состояний, эквивалентных с точки зрения изучаемых свойств. В классическом варианте абстрактная интерпретация проводится по статическому представлению программы в внутрипроцедурном режиме: фиксируется требование монотонности общей передаточной функции подпрограммы, действующей над абстрактными состояниями, а итоговое решение строится как наименьшая или наибольшая неподвижная точка с применением итеративного алгоритма (например, алгоритма Килдалла (Gary Arlen Kildall) [2]). На практике требование монотонности, как правило, выполняется за счет того, что абстрактный домен представляет собой полурешетку, а обновление состояний происходит с применением монотонной относительно частичного порядка полурешетки операции «сбор» или «объединение».

С ходом времени были реализованы системы, воплощающие идею абстрактной интерпретации на практике, в первую очередь с целью поиска ошибок и верификации ПО (например, система Astrée [3]). Однако по большей части такие системы работают на уровне исходных текстов программ и во многом опираются на структурный анализ графа потока управления. В то же время, анализ бинарного кода представляет на настоящий момент не менее, а в каких-то случаях и более востребованное направление. Например, при поиске программных дефектов на уровне исходных текстов Си-программ нет возможности оценить критичность дефекта переполнения буфера, так как то, каким образом компилятор сгенерирует код в таких случаях, не специфицируется стандартом языка (неопределенное поведение).

Экспертиза, накопленная в рамках реализации инструментов анализа полносистемных трасс исполняемого кода, позволяет с уверенностью утверждать, что возможность формализации такого рода задач как задач абстрактной интерпретации позволила бы, как минимум, значительно сократить время экспериментальной фазы исследования новых алгоритмов анализа: без наличия инфраструктуры абстрактной интерпретации невозможно быстро прототипировать такие алгоритмы, так как требуется «с нуля» создавать новый инструмент.

Таким образом, задача построения практически применимой программной системы для проведения произвольно заданной абстрактной интерпретации бинарного кода является актуальной и востребованной. Основные требования к такой системе изложены нами ранее в работе [4]: необходимо (1) промежуточное представление, над которым и будет проводиться абстрактная интерпретация; (2) транслятор бинарного кода в промежуточное представление и (3) сама инфраструктура абстрактной интерпретации. Следует учитывать особенности работы конвейера процессора (слоты задержки, повторяющиеся команды, обработка исключений), наличие механизмов виртуальной памяти (сегментная и страничная трансляция) и т.п. По мнению авторов, одним из наиболее важных практических требований к такой системе становится использование внешних спецификаций для описания семантики выполнения команд и работы конвейера для каждого поддерживаемого процессора.

Работы последних лет [5, 6] в основном сфокусированы на промежуточном представлении и отчасти способе трансляции, сколь-либо выразительная инфраструктура для дальнейшего анализа не предлагается. Во всех случаях предполагается анализ пользовательского кода, работающего в одном пространстве виртуальной памяти. Зачастую рассматриваются только арифметико-логические команды и команды передачи управления, а трансляция происходит программным способом: промежуточное представление строится в коде явно для каждой поддерживаемой команды.

2. Промежуточное представление

Разработанное авторами промежуточное представление Pivot 2 подробно описано в [4]. Его наиболее крупной единицей является модуль, состоящий из следующих элементов.

- **Адресные пространства** единообразно описывают регистровые файлы, память, порты ввода-вывода и т.п. Адресные пространства подразделяются на локальные и удаленные. Локальные адресные пространства локальны для процессорного ядра, в них не возникают исключения при доступе, и тем самым они могут быть представлены как массивы. В удаленных адресных пространствах чтения и записи не имеют таких ограничений, что позволяет моделировать отображенные на память устройства, виртуальную память и т.д.
- **Фрагменты** являются подпрограммами с единственным входом и выходом. Как входными, так и выходными аргументами фрагментов являются битовые векторы, длины которых фиксированы в сигнатурах фрагментов.
- **Базовые блоки** составляют тела фрагментов. Каждый базовый блок состоит из последовательности операторов, а также имеет входные и выходные переменные. При

передаче управления по ребру соответствующие выходные переменные блока-предка копируются во входные переменные блока-потомка. Этот подход является альтернативой использованию φ-функций и позволяет естественным образом задать входы и выходы для фрагментов.

- **Операторы** в базовых блоках описывают выполняемые действия над локальными переменными фрагмента (находящимися в SSA-форме и являющимися битовыми векторами), либо над адресными пространствами. Существует 6 операторов:
  - *APPLY* — применение операции из SMT-логики BV;
  - *CALL* — вызов фрагмента;
  - *INIT* — инициализация переменной константным битовым вектором;
  - *LOAD* — загрузка значения переменной из адресного пространства;
  - *SLICE* — конкатенация и выделение подвекторов из переменных;
  - *STORE* — выгрузка значения переменной в адресное пространство.
- **Разборщики** представляют собой особый вид функций, которые по входному битовому вектору произвольной длины проводят его «лексический разбор», т.е. выдают набор лексем в виде аннотированных битовых векторов. Однако, в отличие от императивно описываемых фрагментов, разборщики задаются декларативно и работают на основе сопоставления образцов (pattern matching).

Операторы *LOAD* и *STORE* учитывают возможность различного порядка байтов в адресном пространстве при доступе, а также предполагают возможность ошибки доступа в случае удаленного адресного пространства.

```
// Адресное пространство регистров, размер адреса — 16 битов.
space regs('16) { x0: '64, x1: '64, … }
// Лексема «мнемоника ADD».
struct add;
// Лексема «операнд-регистр», содержащая номер регистра.
struct reg { rid: '5 }
// Разборщик для команды ADD.
tk rv64i {
    |0000000_nnnnn_mmmmm_000_ddddd_0110011| =>
        add, reg { rid: d }, reg { rid: m }, reg { rid: n };
}
// Семантика команды ADD (для упрощения опущены особенности
// обработки регистра с номером 0).
case fn sema(add: add, rd: reg, rs1: reg, rs2: reg) {
    write_reg(rd, 'add(read_reg(rs1), read_reg(rs2)));
}
fn read_reg(r: reg) -> '64 {
    regs('add(&regs.x0, <r.rid, 0'6>))
}
fn write_reg(r: reg, v: '64) {
    regs('add(&regs.x0, <r.rid, 0'6>)) = v;
}
```

Рис. 1. Пример спецификации машинной команды
Fig. 1. Machine instruction specification example

Кодировки машинных команд, их семантика, а также особенности конвейера могут быть описаны в рамках предложенного представления. Для облегчения задачи написания таких спецификаций был разработан предметно-ориентированный язык и реализован транслятор в промежуточное представление. На рис. 1 представлен фрагмент спецификации команды ADD регистрового сложения 64-разрядного процессора RISC-V. Здесь задается адресное

пространство регистров, лексемы (мнемоники и операнды), разборщик и операционная семантика команды. В общем случае разборщики дополнительно могут параметризоваться режимом работы процессора, если это влияет на декодирование. Фрагменты и разборщики автоматически оптимизируются при трансляции: производится продвижение и свертка констант, частичное удаление избыточности при помощи нумерации значений, частичный вынос вычислений из циклов и упрощение графа потока управления в части удаления пустых блоков и объединения линейных последовательностей блоков.

3. Инфраструктура абстрактной интерпретации

Абстрактная интерпретация проводится в рамках модуля промежуточного представления, и может быть как внутрипроцедурной, так и межпроцедурной. Для проведения абстрактной интерпретации требуется выбрать тип, который будет описывать абстрактное состояние, и задать передаточные функции. Тип абстрактного состояния должен содержать выделенный элемент, соответствующий невозможному состоянию (⊥). Передаточные функции делятся на три группы:

- функции, соответствующие операторам *APPLY*, *CALL*, *INIT*, *LOAD*, *SLICE*, *STORE*, принимают входное состояние и возвращают выходное;
- функция *CALL_RET*, обрабатывающая возврат из вызванного фрагмента;
- функция *EDGE*, обрабатывающая распространение состояний по ребрам ГПУ.

Заданная интерпретация также сообщает инфраструктуре направление обхода: прямое или обратное. Порядок вызова передаточных функций, поведение в части межпроцедурного анализа, а также остановка анализа реализуется инфраструктурной частью, общей для всех разновидностей абстрактной интерпретации, и называемой *исполнителем*. На настоящий момент реализованы два исполнителя.

Исполнитель вдоль пути позволяет применять передаточные функции вдоль некоторого одного пути в программе. При достижении ветвления исполнитель проверяет, что ровно одно из состояний целевых блоков является возможным (т.е. не имеет значение ⊥). Если это так, то исполнитель продолжает продвигаться по пути, а иначе останавливается.

Исполнитель неподвижной точки решает классическую MFP- или LFP-задачу. Он работает в рамках заданного фрагмента, поддерживая текущие состояния на входах и выходах базовых блоков, и обновляет эти состояния в соответствии с алгоритмом, похожим на алгоритм Килдалла [2] (отличия не являются существенными и связаны, в основном, с заменой φ-функций на копирования переменных на ребрах ГПУ). Для этого исполнителя требуется, чтобы состояние являлось полурешеткой с верхним и нижним элементами.

Оба исполнителя при достижении вызова фрагмента могут по желанию пользователя либо продолжить анализ внутри этого фрагмента (с подстановкой текущего состояния), либо использовать предоставленную аннотацию (summary). В первом случае также имеется возможность сохранить аннотацию, полученную при подстановке вызываемого фрагмента, для дальнейшего повторного использования. Вынос реализации межпроцедурного анализа в общую инфраструктурную часть существенно упрощает реализацию самих интерпретаций, т.к. требуется поддерживать абстрактное состояние только одного текущего фрагмента.

На базе разработанной инфраструктуры были реализованы и протестированы:

- **конкретная интерпретация**, эмулирующая выполнение Pivot-операторов и действий на ребрах в рамках фрагмента;
- **символьная интерпретация**, фиксирующая в виде SMT-выражений связи между переменными и ячейками адресных пространств фрагмента и вычисляющая предикаты пути (условия достижимости различных точек в программе);
- **смешанная интерпретация**, являющаяся композицией конкретной и символьной

интерпретации, где только выбранная часть данных анализируется в символьном виде, а оставшиеся данные рассматриваются конкретно.

Кроме того, та же самая инфраструктура абстрактной интерпретации используется и в трансляторе спецификаций в части оптимизаций: например, для частичного удаления избыточности строятся множества доступных выражений в каждой точке фрагмента.

Разделение на непосредственно интерпретацию и исполнители, общие для различных интерпретаций, решает две задачи: во-первых, объем реализации интерпретаций (включая состояние), оказывается весьма мал, а, следовательно, эти реализации проще отлаживать (напомним, что быстрое проведение экспериментов — часть мотивации данной работы); а во-вторых, различные комбинации исполнителей и одних и тех же интерпретаций порождают либо хорошо известные задачи, либо близкие к ним (табл. 1). В таблице также указан объем реализации каждой из интерпретаций на языке Rust.

Табл. 1. Известные задачи как композиция интерпретации и исполнителя
Table 1. Known problems as composition of interpretation and executor

Интерпретация	**Строки кода**	**Исполнитель**	
		Вдоль пути	**Неподвижная точка**
Конкретная	350	эмуляция	свертка и продвижение констант
Символьная	500	«классическое» символьное выполнение по одному пути	слабейшие предусловия сильнейшие постусловия[1]

4. Модель процессора

Для описания поведения процессора используются разборщики и фрагменты задания операционной семантики. Их объединение в единую систему делается на базе модельного конвейера, состоящего из трех этапов: выборка, декодирование, исполнение.

На этапе **выборки** модельный конвейер выполняет действия по определению адресного пространства и адреса в нем очередной команды, а также режима процессора в части, влияющей на декодирование команд. Сюда же относится обработка исключений (в том числе внешних) и таких особенностей как слоты задержки, аппаратные циклы и т.п. Этот этап может быть задан как подпрограмма в спецификации.

На этапе **декодирования** модельный конвейер осуществляет преобразование машинного кода в последовательность лексем. При возникновении ошибки конвейер возвращается к этапу выборки. Декодирование осуществляется по заданным в спецификации разборщикам.

Наконец, этап **выполнения** соответствует анализу фрагмента, задающего семантику декодированной команды. Для этого по сигнатуре (последовательности лексем) выбирается подходящий фрагмент.

Таким образом, каждый полный цикл конвейера может рассматриваться как отображение входного состояния машины в выходное, причем то, какая команда будет выбираться следующей, также заложено в этом преобразовании. Это позволяет, в частности, проводить дизассемблирование методом рекурсивного спуска без дополнительной информации о командах, в том числе параллельно. При анализе по трассам выполнения в последних содержатся сведения об этапах выборки и декодирования, что для части задач позволяет сразу переходить к анализу семантики команд.

[1] В зависимости от выбора направления анализа и задания операций сбора/объединения.

5. Эксперименты

Экспериментальные запуски разработанной и реализованной инфраструктуры проводились на примере Cruehead [7]. Эта программа относится к тестовым программам класса crackme, т.е. ее исследование предполагает обратную инженерию бинарного кода с целью извлечения алгоритма проверки пары «имя пользователь»-«пароль» и генерации такой пары, которая пройдет проверки, заложенные в программе. В [7] описан ручной подход к решению этого примера. В проведенных экспериментах решение строилось автоматически.

Первый эксперимент проводился по предварительно полученной полносистемной трассе выполнения, где было введено произвольное имя пользователя и пароль (некорректная пара). Фрагмент трассы, соответствующий проверке пароля, был проанализирован с помощью смешанной интерпретации и исполнителя вдоль пути. Буфер с паролем был помечен как символьный, и на его байты было установлено дополнительное ограничение: все они должны быть цифрами. Все остальные данные интерпретировались конкретно, и их начальное состояние бралось из трассы. Смешанная интерпретация построила предикат пути, который соответствует корректному паролю (рис. 2). Предикат был отправлен в SMT-решатели Z3 [8] и Boolector [9], в обоих случаях система успешно решилась и был получен корректный пароль.

```
(ne (slice<0, 8> pwd) #x00)
(ne (slice<8, 16> pwd) #x00)
(ne (slice<16, 24> pwd) #x00)
(ne (slice<24, 32> pwd) #x00)
(ne (slice<32, 40> pwd) #x00)
(ne (slice<40, 48> pwd) #x00)
(eq #x000057BB (xor (add (mul (add (mul (add (mul (add (mul (add (mul
  (extu<24> (sub (slice<0, 8> pwd) #x30)) #x0000000A) (extu<24> (sub
  (slice<8, 16> pwd) #x30))) #x0000000A) (extu<24> (sub (slice<16, 24>
  pwd) #x30))) #x0000000A) (extu<24> (sub (slice<24, 32> pwd) #x30)))
  #x0000000A) (extu<24> (sub (slice<32, 40> pwd) #x30))) #x0000000A)
  (extu<24> (sub (slice<40, 48> pwd) #x30))) #x00001234))
```

Рис. 2. Предикат пути в примере Cruehead
Fig. 2. Path predicate in the Cruehead example

Важно отметить, что изображенный на рис. 2 предикат был получен как результат применения предварительных упрощений SMT-выражений. Эти упрощения автоматически производятся разработанной инфраструктурой и позволяют сократить глубину AST-дерева на один десятичный порядок, в основном за счет «склеивания» различных байтов 32-разрядных чисел в единый битовый вектор.

Второй эксперимент проводился по исполнимому образу программы (т.е. ее EXE-файлу). Здесь также применялось смешанное выполнение, но одновременно и имя пользователя, и пароль были помечены как символьные. Было дополнительно установлено символьное ограничение на длину имени пользователя и пароля, для этого в состоянии конкретной части смешанной интерпретации были заданы произвольные строки «имя пользователь» и «пароль». Поскольку в смешанной интерпретации путь определяется по конкретной части состояния, в данном примере этого было достаточно для фиксации длин строк. В данном эксперименте построенная система ограничений также была успешно решена и была получена пара «имя пользователя»-«пароль», которая была принята программой как верная.

Выполнение обоих экспериментов на машине с процессором AMD Ryzen 5 1400 3.20Hz с 32ГиБ оперативной памяти занимает менее секунды, учитывая время загрузки трассы или статического образа, трансляцию машинных команд в промежуточное представление, их анализ и работу двух решателей.

6. Заключение

Данная работа подводит промежуточные результаты продолжающегося исследования. Описаны ключевые особенности разработанной инфраструктуры, обеспечивающие на практике применимость метода абстрактной интерпретации к анализу бинарного кода различных процессорных архитектур. Реализация опробована на двух предварительных экспериментах, которые показали корректную работу.

Дальнейшие планы включают «эргономические» улучшения в предметно-ориентированном языке и его трансляторе (в частности, поддержку тип-параметрических функций и их мономорфизацию), а также в API, которое предоставляет инфраструктура. Планируется реализация в рамках инфраструктуры дизассемблера и легковесного средства трассировки отдельных путей на базе конкретной интерпретации. Набор исполнителей будет дополнен исполнителем по нескольким путям, необходимого для проведения «классического» символьного выполнения. Планируется также подготовка спецификаций для широко распространенных процессорных архитектур. После завершения данных работ планируется этап оценки и улучшения производительности всей системы на корпусе примеров.

Среди первоочередных практических приложений, которые могут быть построены на базе разработанной системы, можно отметить:

- отладчик, основанный на смешанном выполнении, что позволит анализировать отдельные фрагменты кода, когда невозможно запустить анализируемое ПО (так называемый частично работоспособный код);
- средство динамического символьного выполнения, позволяющее определять условия достижимости определенных участков кода;
- средство поиска и ранжирования ошибок и уязвимостей на уровне бинарного кода.

Список литературы / References

[1]. Cousot P, Cousot R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proc. of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, 1977, pp. 238-252.

[2]. Kildall G. A unified approach to global program optimization. In Proc. of the 1st ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, 1973, pp. 194-206.

[3]. Mauborgne L. ASTRÉE: Verification of Absence of Run-Time Error. In Building the information society, IFIP International Federation for Information Processing, vol. 156, 2004, pp. 384-392.

[4]. Соловьев М.А., Бакулин М.Г., Горбачев М.С., Манушин Д.В., Падарян В.А., Панасенко С.С. О новом поколении промежуточных представлений, применяемом для анализа бинарного кода. Труды ИСП РАН, том 30, вып. 6, 2018, стр. 39-68 / Solovev M.A., Bakulin M.G., Gorbachev M.S., Manushin D.V., Padaryan V.A., Panasenko S.S. Next generation intermediate representations for binary code analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 39-68 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-3.

[5]. Dullien T., Porst S. REIL: A platform-independent intermediate representation of disassembled code for static code analysis. In Proc. of the CanSecWest Conference, 2009, 7 p.

[6]. Jung M., Kim S., Han H., Choi J., Cha S.K. B2R2: building an efficient front-end for binary analysis. In Proc. of the NDSS workshop on Binary analysis research, 2019, 10 p.

[7]. Cruehead. URL: https://ansmirnov.ru/crackme-cruehead/ (in Russian), accessed: 23.10.2020.

[8]. The Z3 Theorem Prover. URL: https://github.com/Z3Prover/z3, accessed: 23.10.2020.

[9]. Boolector. URL: https://boolector.github.io/, accessed: 23.10.2020.

Информация об авторах / Information about authors

Михаил Александрович СОЛОВЬЕВ – кандидат физико-математических наук, старший научный сотрудник отдела компиляторных технологий ИСП РАН; старший преподаватель кафедры системного программирования факультета ВМК МГУ. Его научные интересы

включают анализ бинарного и исходного кода, обратную инженерию ПО, операционные системы.

Mikhail Aleksandrovich SOLOVEV is a candidate of physical and mathematical sciences, senior researcher at the compiler technologies department of ISP RAS; senior lecturer at the system programming department of the faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. His research interests include binary and source code analysis, software reverse engineering, and operating systems.

Максим Геннадьевич БАКУЛИН – младший научный сотрудник отдела компиляторных технологий. Его научные интересы включают анализ бинарного и исходного кода, динамический анализ помеченных данных, символьное выполнение, эмуляцию и виртуализацию.

Maksim Gennadevich BAKULIN is a junior researcher at the compiler technologies department. His research interests include binary and source code analysis, dynamic taint analysis, symbolic execution, emulation, and virtualization.

Сергей Сергеевич МАКАРОВ – студент кафедры системного программирования. Его научные интересы включают анализ бинарного кода, эмуляцию и виртуализацию, операционные системы, обратную инженерию ПО.

Sergei Sergeevich MAKAROV is a student at the system programming department. His research interests include binary code analysis, emulation and virtualization, operating systems, and software reverse engineering.

Дмитрий Валерьевич МАНУШИН – аспирант кафедры системного программирования. Его научные интересы включают анализ бинарного кода, анализ исходного кода, безопасность ПО.

Dmitrii Valerevich MANUSHIN is a postgraduate at the system programming department. His research interests include binary code analysis, source code analysis and software security.

Вартан Андроникович ПАДАРЯН – кандидат физико-математических наук, ведущий научный сотрудник отдела компиляторных технологий ИСП РАН; доцент кафедры системного программирования факультета ВМК МГУ. Его научные интересы включают компиляторные технологии, безопасность ПО, анализ бинарного кода, параллельное программирование, эмуляция и виртуализация.

Vartan Andronikovich PADARYAN is a candidate of physical and mathematical sciences, leading researcher at the compiler technologies department of ISP RAS; associate professor of the system programming department of the faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. His research interests include compiler technologies, software security, binary code analysis, parallel programming, emulation, and virtualization.

DOI: 10.15514/ISPRAS–2020–32(6)–9

Способ маскирования передаваемой информации

[1] *П.В. Закалкин, ORCID: 0000-0003-2946-2586 <pzakalkin@mail.ru>*
[1] *С.А. Иванов, ORCID: 0000-0002-0528-276X <sa-ivanov@inbox.ru>*
[1] *Е.В. Вершенник, ORCID: 0000-0003-3647-741X <sukhorukova_lena@mail.ru>*
[2] *А.В. Кирьянов, ORCID: 0000-0001-6104-7433 <alex1175@mail.ru>*

[1] *Военная академия связи им. С.М. Буденного,*
194064, Россия, Санкт-Петербург, Тихорецкий проспект, д. 3
[2] *Академия Федеральной Службы охраны Российской Федерации,*
302015, Россия, г. Орел, ул. Приборостроительная, д. 35

Аннотация. Процессы глобализации, появление и активное развитие киберпространства привели к необходимости защиты информации, передаваемой в рамках информационного обмена. Существующие подходы к защите информации, а в частности ее шифрование, стеганография и т.п. с точки зрения информационного обмена имеют ряд демаскирующих признаков, которые, при всей несомненной надежности этих подходов, существенно снижают скрытность передачи информации. Предлагаемая статья рассматривает подход, позволяющий осуществлять скрытую передачу защищаемой информации по открытым каналам связи, за счет маскирования передаваемой информации. Разработка предлагаемого подхода осуществлялась поэтапно, на первом этапе был разработан и запатентован способ маскирования передаваемой информации. На следующем этапе, на основе разработанного способа, создана функциональная модель клиентского и серверного приложений комплекса передачи скрытой информации. Передачу маскированной информации предлагается осуществлять с помощью разработанного протокола скрытой передачи информации. Структурная схема пакета предлагаемого протокола передачи скрытой информации, вариант реализации и использования протокола на прикладном уровне представлены в работе. На заключительном этапе, разработана программная реализация предлагаемого подхода и осуществлено моделирование информационного обмена при различном окне смещения. В работе представлена функциональная модель разработанного комплекса, схема взаимодействия функциональных модулей, и блок-схема предлагаемого подхода к маскированию передаваемой информации. Повышение скрытности передачи информации обеспечивается за счет процедуры преобразования несущего сообщения в маркерное путем формирования окна смещения, а также использования массива цифровых записей для выбора несущего сообщения. Предлагаемый подход позволяет при увеличении размера окна и использовании скользящего окна использовать меньшее несущее сообщение, в зависимости от размера информационного сообщения выбирать оптимальный размер несущего сообщения и окна смещения.

Ключевые слова: маскирование информации; скрытая передача

Для цитирования: Закалкин П.В., Иванов С.А., Вершенник Е.В., Кирьянов А.В. Способ маскирования передаваемой информации. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 111-126. DOI: 10.15514/ISPRAS–2020–32(6)–9

Method of Masking Transmitted Information

[1] *P.V. Zakalkin, ORCID: 0000-0003-2946-2586 <pzakalkin@mail.ru>*
[1] *S.A. Ivanov, ORCID: 0000-0002-0528-276X <sa-ivanov@inbox.ru>*
[1] *E.V. Vershennik , ORCID: 0000-0003-3647-741X <sukhorukova_lena@mail.ru>*
[2] *A.V. Kir'yanov, ORCID: 0000-0001-6104-7433 <sa-ivanov@inbox.ru>*

[1] *Military Telecommunications Academy,*
3, Tikhoretsky pr., Saint-Peterburg, 194064, Russia
[2] *Academy of the Federal Guard Service of the Russian Federation,*
35, Priborostroitelnaia st., Orel, 302015, Russia

Abstract. The processes of globalization, the emergence and active development of cyberspace have led to the need to protect information transmitted in the framework of information exchange. Existing approaches to information protection, in particular its encryption, steganography, etc. from the point of view of information exchange have a number of unmasking features that, despite the undoubted reliability of these approaches, significantly reduce the secrecy of information transmission. The proposed article considers an approach that allows for the hidden transmission of protected information over open communication channels, by masking the transmitted information. The development of the proposed approach was carried out in stages. at the first stage, a method for masking the transmitted information was developed and patented. At the next stage, on the basis of the developed method, a functional model of client and server applications of the hidden information transmission complex is created. The transfer of masked information is proposed to be carried out using the developed protocol of hidden information transfer. The block diagram of the package of the proposed Protocol for transmitting hidden information, the implementation and use of the Protocol at the application level are presented in this paper. At the final stage, a software implementation of the proposed approach was developed and modeling of information exchange at different offset Windows was performed. the paper presents a functional model of the developed complex, a scheme of interaction of functional modules, and a block diagram of the proposed approach to masking the transmitted information. Increasing the secrecy of information transmission is provided by the procedure for converting a carrier message into a marker message by forming an offset window, as well as using an array of digital records to select the carrier message. The proposed approach allows you to use a smaller carrier message when increasing the window size and using a sliding window depending on the size of the information message, you can choose the optimal size of the carrier message and the offset window.

Keywords: concealment of information; hidden transfer

For citation: Zakalkin P.V., Ivanov S.A., Vershennik E.V., Kir'yanov A.V. Method of masking transmitted information. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 111-126 (in Russian). DOI: 10.15514/ISPRAS-2020-32(6)-9

1. Введение

Интеграция сетей связи различных операторов с Единой сетью электросвязи Российской Федерации (ЕСЭ РФ), наряду со значительными удобствами и удешевлением процесса организации связи (нет затрат на строительство и обслуживание линий связи), способствовало тому, что нарушители активно применяют различные средства информационно-технических воздействий.

Защищать передаваемую (или хранимую) информацию от несанкционированного использования приходится во многих случаях. Это требуется при решении государственных, дипломатических, военных задач, в работе бизнеса (коммерции), при исследовании новых научно-технических проблем, при разработке оригинальных технологических процессов и устройств. Защищать информацию требуется при документообороте в государственных организациях и при ведении частной переписки. Современные телекоммуникационные технологии невозможно представить без защиты передаваемой информации [1-2].

Основным подходом к защите передаваемой информации является ее шифрование. Данный подход считается наиболее приемлемым и позволяет скрыть передаваемую информацию.

Более того, порядка 70-80 % (из которого 90 % трафик https) мирового трафика передается в зашифрованном виде и потенциальному нарушителю, для осуществления какого-либо воздействия, среди этого трафика необходимо найти нужный.

На первый взгляд подход с шифрованием информации не лишен недостатков, однако:

- крупные корпоративные структуры, банковская сфера, элементы критической инфраструктуры используют специализированные программно-аппаратные средства шифрования трафика (например, криптомаршрутизаторы);
- взлом и дальнейший анализ трафика HTTPS для высококвалифицированного нарушителя не составляет особого труда;
- нарушитель будет анализировать трафик, исходящий от специализированных программно-аппаратных средств.

Для высококвалифицированного нарушителя, проводящего целевые атаки, сам факт наличия аппаратуры, предусматривающей шифрование, является мощным демаскирующим признаком. При этом, тип применяемого средства шифрования и используемый им алгоритм определить не сложно – достаточно проанализировать служебный трафик при установлении соединения.

Потенциальный нарушитель будет стремится получить доступ к защищаемой информации или вывести из строя закрытые каналы связи (для предотвращения информационного обмена). Таким образом, само наличие закрытого канала связи является своеобразным сигналом того, что в нем с большой долей вероятности передается защищаемая информация. При всем этом, для обеспечения информационного взаимодействия крупных территориально разнесенных корпоративных структур, банковской сферы и т.д. используется множество закрытых каналов передачи информации, по которым могут передаваться сведения, содержащие как государственную и коммерческую тайну, так и конфиденциальную информацию [3-4].

Логичным выходом из сложившейся ситуации является скрытая передача защищаемой информации по открытым каналам связи. При этом наличие нарушителя накладывает некоторые ограничения в виде необходимости передачи трафика в открытом виде и по одному каналу связи, который потенциально может контролироваться нарушителем. В связи с этим возникает необходимость разработки и исследования новых систем со скрытой передачей информации.

2. Способы скрытой передачи информации

Для решения поставленной задачи в первую очередь была рассмотрена стеганография. Данный подход позволяет разместить в цифровых файлах-контейнерах достаточно большой объем информации без явного искажения изображения, наличием априорных сведений о размерах контейнера, существованием в большинстве реальных изображений текстурных областей, имеющих шумовую структуру и хорошо подходящих для встраивания информации, проработанностью способов цифровой обработки изображений и цифровых форматов представления изображений [5-8]. Однако существенным недостатком является невозможность обеспечения скрытного информационного обмена в силу внесения изменений (преобразований) в цифровой файл-контейнер, которые могут быть детектированы с помощью известных программных продуктов стеганоанализа.

После этого были рассмотрены альтернативные подходы к скрытному обмену информацией [9-12]. Однако они либо использовали подходы основанных на стеганографии, либо требовали дополнительного оборудования и обладали не высокой скрытностью.

Наиболее подходящим к сложившимся условиям и наложенным ограничениям является «Способ скрытой передачи информации» Патент РФ № 2552145, однако и он обладает рядом недостатков. Основными из которых можно выделить: необходимость использования

нескольких каналов связи; в качестве несущего изображения возможно использовать только одно изображение; низкая информационная емкость маркерного сообщения; отсутствие возможности изменения окна смещения. Данные недостатки существенно снижают скрытность передачи информации за счет достаточно простых действий при скрытии информационного сообщения, но являются устранимыми.

3. Разработка функциональной модели программного комплекса маскирования передаваемой информации

В рамках решения этой проблемы была разработана функциональная модель и запатентован способ маскирования передаваемой информации [13], на основе которого был разработан программный комплекс [14]. На рис. 1, 2 представлена функциональная модель программного комплекса, отображающая особенности его функционирования и внутреннюю взаимосвязь элементов.

Прежде чем приступить к дальнейшему рассмотрению предлагаемого способа необходимо дать определения используемых терминов.

1) Информационное сообщение – сообщение, предназначенное для передачи абоненту и являющееся цифровой записью в двоичном виде.
2) Несущее сообщение – сообщение, которое представляет собой цифровую запись в двоичном виде, являющуюся реальными файлами соответствующих расширений. Например, для аудио файлов расширения: .mp3, .ac3, .3ga и т.д., для видеофайлов .mp4, .mpg, .mkv и т.д.
3) Маркерное сообщение – цифровая запись в двоичном виде, полученная после маскирования информационного сообщения с помощью несущего сообщения и предназначенная для передачи по каналу связи.
4) Маркерный пакет – пакет, передаваемый по каналу связи и включающий в себя заголовок и маркерное сообщение.

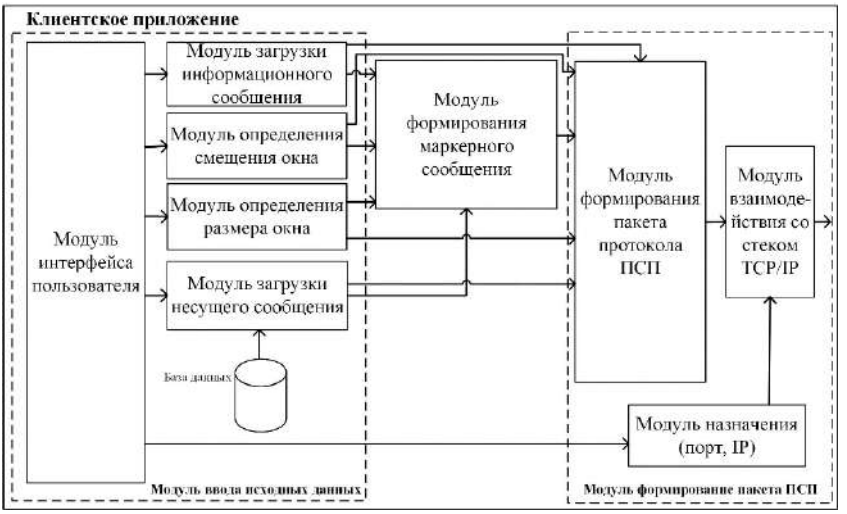

Рис. 1. Функциональная модель клиентского приложения разработанного программного комплекса передачи скрытой информации

Fig. 1. Functional model of a client application developed by a software package for transmitting hidden information

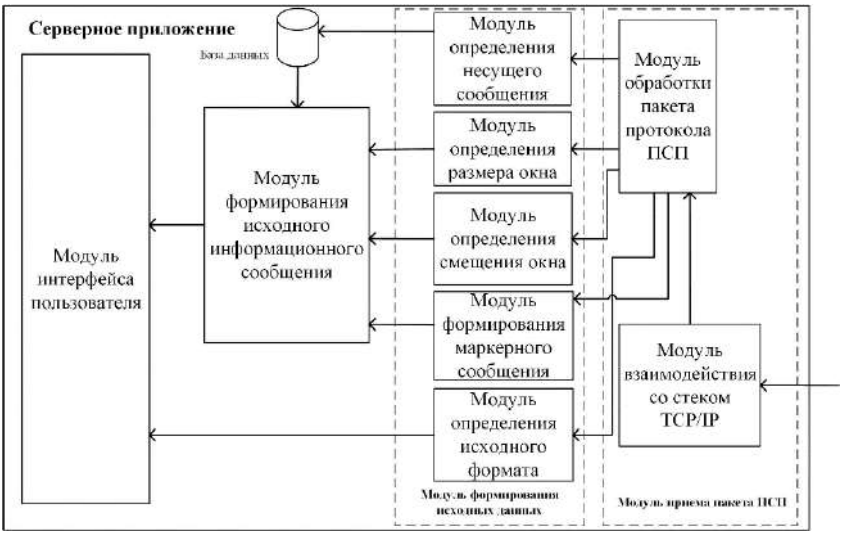

Рис. 2. Функциональная модель серверного приложения разработанного программного комплекса передачи скрытой информации
Fig. 2. Functional model of a server application developed by a software package for transmitting hidden information

Разработка функциональной модели является одним из основных этапов проектирования систем и предназначена для изучения особенностей работы системы до её программной, аппаратной или программно-аппаратной реализации. Функциональная модель является абстрактной моделью создаваемой системы и содержит все необходимые элементы разрабатываемой системы, а также их взаимосвязь и взаимодействие как между собой, так и с окружающими объектами.

Разработанная функциональная модель включает 2 основных модуля:

- серверное приложение;
- клиентское приложение.

«Модуль взаимодействия со стеком TCP/IP» предназначен для приема всего трафика от клиентского приложения, выделения метки времени, а также выделения информации об адресе отправителя и номере порта для каждого пакета и дальнейшей передачи пакета на соответствующий модуль.

«Модуль обработки пакета протокола скрытой передачи информации (ПСП)» принимает информацию от модуля взаимодействия со стеком TCP/IP, а также осуществляет распределение информации на модули формирования маркерного сообщения, определения размера окна, определения смещение окна, определения несущего сообщения, определения исходного формата.

«Модуль формирование маркерного сообщения» принимает последовательность бит с модуля обработки пакета ПСП, выстраивает их в правильной последовательности и осуществляет ее передачу на модуль формирования исходного информационного сообщения.

«Модуль определения размера окна» принимает данные с модуля обработки пакета ПСП. В данном модуле собирается характеристика о размере окна, которая была задана в клиентском приложении для формирования маркерного сообщения.

«Модуль определения смещения окна» принимает данные с модуля обработки пакета ПСП. В данном модуле собирается характеристика о смещении окна, которая была задана в клиентском приложении для формирования маркерного сообщения.

«Модуль определения исходного формата» принимает информацию с модуля обработки пакета ПСП. В данном модуле определяется информация о формате переданного сообщения с клиентского приложения, например, doc, jpeg и т.п.

«Модуль определения несущего сообщения» принимает информацию с модуля обработки пакета ПСП. В данном модуле определяется информация о номере несущего сообщения, хранящегося в базе данных. Информация о несущем сообщении нужна для того, чтобы получить правильное информационное сообщение.

«Модуль формирования исходного информационного сообщения» принимает данные от модулей формирования маркерного сообщения, определения смещения окна, определения размера окна, определения несущего сообщения и с базы данных. В результате применения алгоритма извлечения информационного сообщения из маркерного мы получаем исходное сообщение, которое передавалось с клиентского приложения.

«Модуль формирования информационного сообщения» принимает данные с модуля формирования исходного информационного сообщения. В результате мы получаем файл, который передавался с клиентского приложения.

«Модуль интерфейса пользователя» предназначен для осуществления взаимодействия с пользователем и установки исходных данных, а также для последующего формирования кадров, необходимых для настройки модуля взаимодействия клиент/сервера.

«База данных» содержит множество различных файлов, с помощью которых в алгоритме сокрытия конфиденциальной информации формируется несущее сообщение.

«Модуль назначения» отвечает за передачу на модуль взаимодействия со стеком TCP/IP таких характеристик, как номер порта назначения и IP адрес.

Схема взаимодействия функциональных модулей представлена на рис. 3.

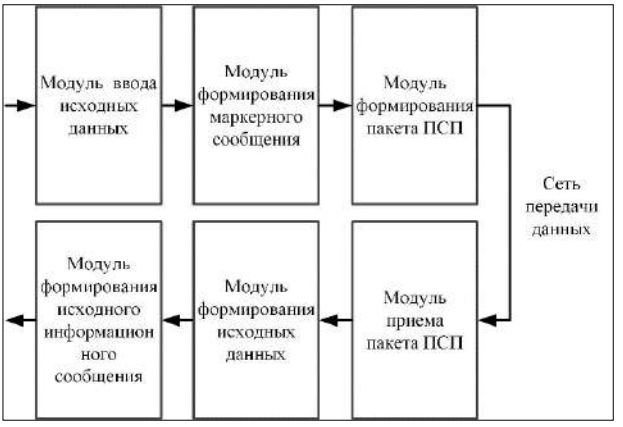

Рис. 3. Схема взаимодействия функциональных модулей
Fig. 3. Scheme of interaction of functional modules

Клиентское приложение в свою очередь состоит из следующих модулей:

- модуль взаимодействия со стеком TCP/IP;
- модуль обработки пакета ПСС;
- модуль интерфейса пользователя;
- модуль загрузки информационного сообщения;
- модуль определения размера окна;
- модуль загрузки несущего сообщения;
- модуль формирование маркерного сообщения;
- модуль определения смещения окна;
- модуль назначения;

- база данных.

Несмотря на некоторую тривиальность отдельных блоков, для удобства восприятия блок-схема предлагаемого способа маскирования информации (рис. 4.) представлена в более подробном виде.

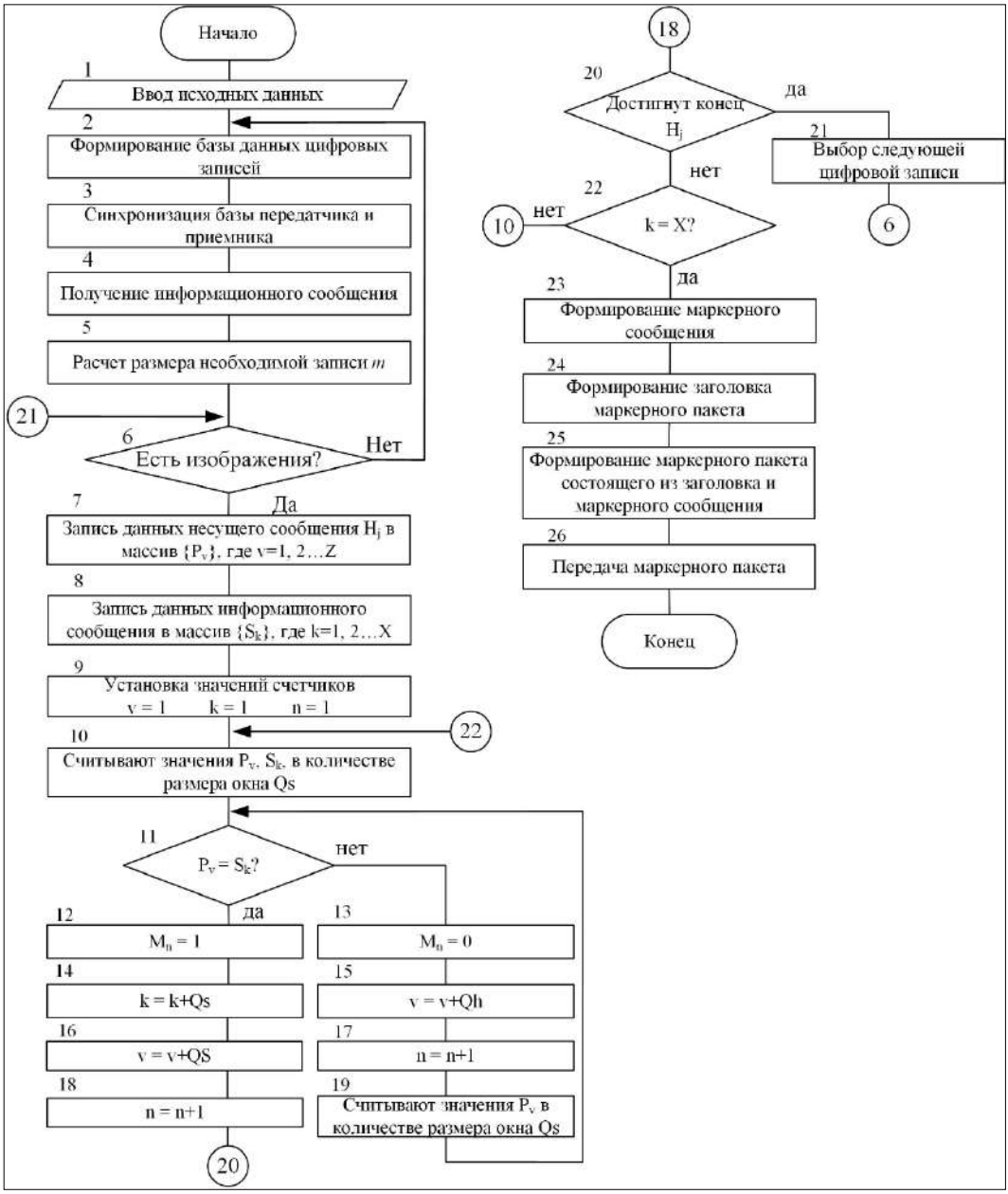

Рис. 4. Блок-схема способа маскирования передаваемой информации
Fig. 4. Block diagram of the method for masking transmitted information

На первоначальном этапе маскирования передаваемой информации осуществляют задание исходных данных:

Qh – значение маркера смещения несущего сообщения;

Qs – размер окна смещения информационного сообщения.

Кроме того, формируют массивы для запоминания битовой последовательности информационного сообщения $\{S_k\}$, где $k = 1, 2, \dots, X$, битовой последовательности несущего сообщения $\{P_v\}$, где $v = 1,2, \dots, Z$, битовой последовательности маркерного сообщения $\{M_n\}$, где $n = 1, 2, \dots, Y$.

После этого осуществляют формирование базы данных цифровых записей в виде видео-, аудио-файлов (информация) и т.д. База данных цифровых записей (несущих сообщений) $\{B\}$ состоит из массива несущих сообщений $\{H_j\}$; в свою очередь, несущее сообщение состоит из последовательности бит $\{h_v\}$.

Далее синхронизируют базу передающего и принимающего абонента. В процессе синхронизации сравниваются все цифровые записи в базах данных передающего и принимающего абонентов. При необходимости осуществляется дополнение цифровых записей в базах для достижения полной идентичности баз данных. Базы считаются синхронизированными, если все цифровые записи в базах данных отправителя и получателя совпадают.

Создание базы данных и входящих в нее цифровых записей осуществляется по предварительной договоренности корреспондентов любым удобным для них способом. Например, возможен обмен базами данных на flash, CD и т.д. носителях. Возможно скачивание цифровых записей из социальных сетей, со сторонних ресурсов, например, FTP сервера содержащие исходные тексты программного обеспечения, содержат множество файлов различных форматов (при этом все они имеют контрольные суммы) и т.д. Таким образом, даже при полном контроле трафика потенциальным злоумышленником процесс формирования и синхронизации базы данных будет выглядеть как обычная активность пользователей в офисе (просмотр соцсетей, e-mail, загрузка аудио и видео файлов и т.д.).

Информационное сообщение, которое необходимо передать, представляется в цифровом виде, позволяющем в блоке 9 (рис. 4) записать его в массив $\{S_k\}$, где $k = 1, 2, \dots, X$.

Для осуществления передачи информационного сообщения рассчитывают размер необходимой цифровой записи, которая далее будет использоваться как несущее сообщение H_j. Учитывая размеры информационного и несущего сообщений (при большом количестве испытаний), в соответствии с предельной теоремой, суммарный закон распределения случайных величин будет соответствовать нормальному закону. Таким образом, для определения объема несущего сообщения, необходимого для переноса информационного сообщения используется выражение:

$$P_v \geq S_n \cdot 2^{Qs},$$

где P_v – массив, содержащий битовую последовательность несущего сообщения; S_n – массив, содержащий битовую последовательность информационного сообщения; Qs – размер окна смещения для информационного сообщения.

Из базы данных выбирают цифровую запись, превышающую по размеру рассчитанное значение несущего сообщения, но при этом наиболее близкую по размеру. Выбранная таким образом цифровая запись будет иметь превышение размера для обеспечения преобразования текущего информационного сообщения. В случае, если цифровая запись необходимого размера в базе данных не обнаружена, то формируют дополнительные цифровые записи и записывают их в базу данных цифровых записей с учетом рассчитанного размера несущего сообщения.

Битовые последовательности несущего сообщения H_j записывают в массив $\{P_v\}$, где $v = 1,2, \dots, Z$, а битовые последовательности информационного сообщения записывают в массив $\{S_k\}$, где $k = 1, 2, \dots, X$.

После этого последовательно считывают из соответствующих массивов значения P_v и S_k в количестве, равном размеру окна смещения Qs и осуществляют сравнение считанных

битовых последовательностей несущего и информационного сообщений P_v и S_k. При совпадении данных битовых последовательностей флаговое значение M_n устанавливают в «единицу», в противном случае флаговое значение M_n устанавливают в «ноль» (рис. 5).

Рис.5. Схема, поясняющая процесс формирования маркерного сообщения
Fig.5. Diagram explaining the process of forming a marker message

Значение счетчика k увеличивают на размер окна смещения для информационного сообщения Qs:

$$k = k + Qs.$$

Это соответствует переходу и анализу следующих по порядку битовых последовательностей информационного сообщения, сдвинутых на размер окна смещения информационного сообщения.

Значение счетчика v увеличивают на значение маркера смещения для несущего сообщения Qh:

$$v = v + Qh.$$

Это соответствует переходу и анализу следующих по порядку битовых последовательностей информационного сообщения, сдвинутых на значение маркера смещения несущего сообщения.

Значение счетчика v увеличивают на размер окна смещения для информационного сообщения Qs:

$$v = v + Qs.$$

Значение счетчика n увеличивают на «единицу» $n = n + 1$ для формирования следующих битовых значений маркерного сообщения.

После чего считывают значения следующей битовой последовательности P_v в количестве размера окна Qs и переходят к блоку 12 (рис. 4), где сравнивают битовые последовательности несущего и информационного сообщений P_v и S_k.

В случае достижения окончания несущего сообщения из базы данных цифровых записей в качестве несущего сообщения выбирают цифровую запись, превышающую по размеру текущую, но при этом наиболее близкую к ней по размеру.

При достижении окончания информационного сообщения формируют маркерное сообщение, представляющее собой битовую последовательность Mn, состоящее из записанных флаговых значений «ноль» и «единица», которое было сформировано в блоках 12 и 13 (рис. 4).

Для передачи маркерного пакета по каналу связи формируют заголовок маркерного пакета, состоящий из номеров выбранных цифровых записей, размера окна смещения Qs и значения маркера смещения Qh. После чего формируют маркерный пакет, состоящий из заголовка и маркерного сообщения, и записывают его в информационное поле пакета передаваемого по сети связи.

Передачу маркерного сообщения предлагается осуществлять с помощью протокола скрытой передачи информации, структурная схема пакета которого представлена на рис. 6.

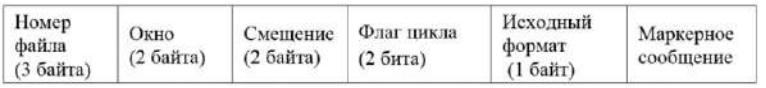

Рис. 6. Структурная схема пакета протокола передачи скрытой информации
Fig. 6. Block diagram of the hidden information transfer Protocol package

Структурная схема пакета протокола скрытой передачи информации предусматривает следующие поля для данного протокола: номер файла; размер окна; размер смещения; флаг цикла; исходный формат; маркерное сообщение.

- Номер файла – поле, предназначенное для передачи информации о примененном файле несущего сообщения. Размер поля 3 байта.
- Размер окна – поле, показывающее, какое количество бит сравнивается в алгоритме сокрытия информации. Размер поля 2 байта.
- Смещение – поле, в котором указывается количество бит смещения при формировании маркерного сообщения. Размер поля 2 байта.
- Флаг цикла – поле, которое показывает, что применялось циклическое использование одного несущего сообщения. Размер поля 2 бита.
- Исходный формат – поле, которое указывает, в каком исходном формате было информационное сообщение (doc, jpeg, и т.п.). Размер поля 1 байт. Обозначение форматов для заполнения поля представлено в табл. 1.

Табл. 1. Обозначение форматов в поле исходный формат
Table. 1. The designation of the formats in the original format

Формат	**Обозначение**	**Формат**	**Обозначение**
.jpg	00000000	.png	00001001
.docx	00000001	.ico	00001010
.mp4	00000010	.bmp	00001011
.gif	00000011	.xmcd	00001100
.pdf	00000100	.log	00001101
.vsd	00000101	.db	00001110
.txt	00000110	.md	00001111
.doc	00000111	.fb2	00010000
.pptx	00001000		

Наличие нестандартного протокола передачи (протокол скрытой передачи информации) будет серьезным демаскирующим признаком для системы связи любой организации его использующей. Однако, принимая во внимание принцип инкапсуляции протоколов,

предлагается использовать протокол не на сетевом или транспортном уровне, а на прикладном (рис. 7). Это позволит использовать любой протокол прикладного уровня для переноса в поле данных вложенного протокола передачи скрытой информации.

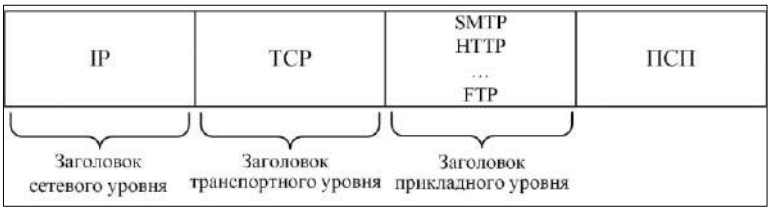

Рис. 7. Графическое отображение инкапсуляции пакета протокола скрытой передачи информации на примере стека протоколов TCP/IP
Fig. 7. Graphical representation of the encapsulation of a packet of the hidden information transfer protocol on the example of the TCP / IP Protocol stack

При анализе сетевого трафика злоумышленник будет видеть TCP/IP пакеты в незашифрованном виде, при этом параметры трафика будут абсолютно не отличимыми от стандартного трафика при любом другом информационном обмене. Учитывая большие объемы открытого трафика и отсутствие какой-либо априорной информации о том, с кем идет информационный обмен и что необходимо искать в передаваемом трафике (протоколы прикладного уровня), задача найти TCP/IP пакеты, содержащие в себе пакеты протокола скрытой передачи информации сравнима с задачей поиска иголки в стоге сена.

Для моделирования размеров передаваемых файлов, типовых сообщений, Web-страниц часто используется распределение Парето, которое имеет следующий вид

$$w(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}}, \alpha > 0, k > 0, x > 0, \quad (1)$$

Где α – параметр формы; k – параметр, определяющий нижнюю границу для случайной величины.

С помощью ЭВМ возможно генерировать равномерно распределенную случайную величину y в диапазоне от 0 до 1. Для того чтобы моделировать случайную величину x с плотностью распределения вероятности Парето, необходимо найти функциональное преобразование $x = f(y)$.

Это можно сделать на основе выражения

$$w\big(x = f(y)\big) = w(y)\left|\frac{dy}{dx}\right|, \quad (2)$$

откуда

$$y = \int w(x)dx = \int \frac{\alpha k^{\alpha}}{x^{\alpha+1}}dx = 1 - \left(\frac{k}{x}\right)^{\alpha}, \quad (3)$$

при $x > k$, $\alpha > 0$.

Таким образом, получаем следующее выражение для моделирования случайной величины с плотностью распределения вероятности Парето:

$$x = \frac{k}{\sqrt[\alpha]{1 - rand}}, \quad (4)$$

где $rand$ – случайное число, подчиненное равномерному закону распределения, генерируемое на ЭВМ в диапазоне от 0 до 1. Параметр α связан с показателем Херста выражением $\alpha = 3 - 2H$.

Параметр α обычно вычисляется на основе метода максимального правдоподобия по известным результатам измерения интенсивности реального трафика $X = [x_1, x_2, \dots, x_n]$:

$$\alpha = \frac{n - 1}{\sum_{i=1}^{n} logX_i - nlogk}, \quad (5)$$

где $k = \min_i k$.

На практике значение параметра самоподобия находится в промежутке от 1\2 до 3\2. Таким образом, можно осуществить моделирование размеров как несущих сообщений, так и информационных сообщений. Более важную роль будет играть отношение размера несущего сообщения к информационному сообщению $(L_{НС}/L_{ИС})$.

Из логики функционирования способа следует, что при малом несущем сообщении и большом информационном сообщении, алгоритм не будет сходиться. Поэтому необходимо определить $G_{ДОП} > (L_{НС}/L_{ИС})$ – границу, при которой размера несущего сообщения будет достаточно для нахождения отображения в нем информационного сообщения, либо рассмотреть вариант формирования последовательности нескольких несущих сообщений для переноса информации одного сообщения.

Так как появление в сообщениях битов «0» и «1» подчиняется равномерному закону, то можно сделать предположение о том, что композиция двух законов равномерной плотности распределения, заданных на одном и том же участке, будет соответствовать закону распределения Симпсона (иначе «закон треугольника») [15].

Однако, учитывая размеры сообщений $L_{ИС}$ и $L_{НС}$ (при большом количестве испытаний), в соответствии с предельной теоремой, суммарный закон распределения случайных величин будет соответствовать нормальному закону.

В ходе исследования и практической реализации способа получены результаты, подтверждающие гипотезу. Количество испытаний для каждого размера окна $N_{исп} = 20000$. На рис. 8 показана зависимость частоты совпадений отношения объема информационного сообщения к объему маркерного сообщения.

Таким образом, для определения объема несущего сообщения $L_{НС}$, необходимого для переноса информационного сообщения $L_{ИС}$, было получено выражение

$$L_{НС} \geq L_{ИС} \cdot 2^{L_{окна}}.$$

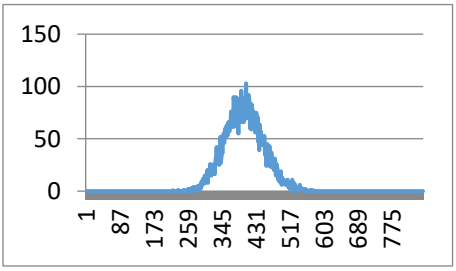

Рис. 8. Зависимость частоты совпадений отношения объема информационного сообщения к объему маркерного сообщения (Lокна = 2, Vинф = 200)

Fig. 8. Dependence of the frequency of matches of the ratio of the volume of the information message to the volume of the marker message (Lокна = 2, Vинф = 2000)

В табл. 2 представлены статистические данные соответствия размера информационного сообщения размеру несущего сообщения при величине окна сравнения 2 и 3 бита.

Табл. 2. Результаты моделирования при смещении окна на $L_{окна}$

Table. 2.– Simulation results when the window is shifted by $L_{окна}$

$L_{ИС}$	$L_{окна}$	$L_{МС}$	$L_{НС}$	$L_{окна}$	$L_{МС}$	$L_{НС}$
100	2	200	400	3	267	800
200	2	400	800	3	533	1600
300	2	600	1200	3	800	2400
400	2	800	1600	3	1067	3200
500	2	1000	2000	3	1333	4000

Дальнейшее исследование проводилось в направлении реализации процедуры скользящего окна. При реализации алгоритма со скользящим окном, смещение окна производится не на размер окна, а на один бит. При этом размер маркерного сообщения увеличивается – каждому биту несущего сообщения ставится в соответствие бит маркерного сообщения. Информационная емкость несущего сообщения увеличивается. Результаты моделирования представлены в табл. 3.

Табл. 3. Результаты моделирования при смещении окна на 1 бит (скользящее окно)
Table. 3. Simulation results when the window is shifted by 1 bit (sliding window)

$L_{ИС}$	$L_{окна}$	$L_{МС}$	$L_{НС}$	$L_{окна}$	$L_{МС}$	$L_{НС}$	$L_{окна}$	$L_{МС}$	$L_{НС}$
100	2	179	179	3	278	278	4	391	391
200	2	387	387	3	530	530	4	785	785
300	2	591	591	3	786	786	4	1189	1189
400	2	795	795	3	1078	1078	4	1570	1570
500	2	989	989	3	1317	1317	4	1934	1934

Таким образом, предлагаемый способ позволяет при увеличении размера окна и использовании скользящего окна использовать меньшее несущее сообщение. Помимо этого, в зависимости от размера информационного сообщения выбирать оптимальный размер несущего сообщения и окна смещения.

Необходимо понимать, что увеличивать размер окна до бесконечности нельзя и при многократном увеличении размера окна, произойдет кардинально противоположная ситуация и размер несущего сообщения будет несоизмеримо больше размера информационного сообщения.

Основными этапами функционирования разработанного программного комплекса [14] являются следующие:

1) выполняется считывание входных параметров, на основе которых строится вся дальнейшая работа алгоритма;
2) исходное сообщение преобразовывается в маркерное сообщение;
3) устанавливается соединение клиента и сервера, осуществляется синхронизация баз данных;
4) сообщения делятся на пакеты и передаются в зашифрованном виде по каналу связи;
5) принимается последний пакет и осуществляется разъединение между клиентом и сервером;
6) из принятых пакетов извлекаются основные параметры алгоритма маскирования передаваемой информации;
7) маркерное сообщение преобразуется в исходное сообщение;
8) полученное сообщение сохраняется в удобном формате, исходя из требований пользователя.

Основными функциональными возможностями программного комплекса являются:

- передача сообщения в скрытом виде;
- выбор режима (автоматически, самостоятельно) изменения основных параметров алгоритма сокрытия информации, таких как размер окна и смещение окна;
- возможность добавления различных узлов;
- передача текста, цифровых записей различного формата.

4. Заключение

Представленные в работе исследования, посвященные разработке способа маскирования передаваемой информации [13], программного комплекса [14] и протокола скрытой передачи

информации позволяют осуществлять передачу защищаемой информации по открытым каналам связи, а также, в зависимости от размера передаваемого информационного сообщения, можно выбирать оптимальный размер окна и его смещение.

Список литературы / References

[1]. Цыцулин А.К., Зубакин И.А. Концепция качества информации в теории связи. Вопросы радиоэлектронники. Серия: Техника телевидения, № 4, 2016 г. стр. 19-25 / Tsyculin A.K., Zubakin I.A. Quality information concept in the theory of communication. Voprosy radiojelektronniki. Serija: tehnika televidenija, № 4, 2016, pp. 19-25 (in Russian).

[2]. Коцыняк М.А., Кулешов И.А., Кудрявцев А.М., Лаута О.С. Киберустойчивость информационно-телекоммуникационной сети. СПб., Бостон-спектр, 2015 г., 150 стр. / Kotsynjak M.A., Kuleshov I.A., Kudrjavtsev A.M., Lauta O.S. Cyber stability of the information and telecommunications network. SPb., Boston-spektr, 2015, 150 p. (in Russian).

[3]. Бегаев А.Н., Гречишников Е.В., Добрышин М.М., Закалкин П.В. Предложение по оценке способности узла компьютерной сети функционировать в условиях информационно-технических воздействий. Вопросы кибербезопасности, № 3(27), 2018 г., стр. 2-8. / Begaev A.N., Grechishnikov E.V., Dobryshin M.M., Zakalkin P.V. Proposal for assessing the ability of a computer network node to function in conditions of information and technical influences. Voprosy kiberbezopasnosti, № 3 (27), 2018, pp. 2-8(in Russian).

[4]. Гречишников Е.В., Добрышин М.М., Закалкин П.В. Модель узла доступа VPN как объекта сетевой и потоковой компьютерных разведок и DDOS-атак. Вопросы кибербезопасности, № 3(16), 2016 г., стр. 4-12 / Grechishnikov E.V., Dobryshin M.M., Zakalkin P.V. A model of a VPN access point as of an object of network and streaming computer intelligence and DDOS attacks. Voprosy kiberbezopasnosti, № 3(16), 2016, pp.4-12(in Russian).

[5]. Цветков К.Ю., Федосеев В.Е., Коровин В.М., Абазина Е.С. Способ скрытой передачи данных в видеоизображении. Патент на изобретение RUS 2608150, опубл. 16.01.2017, 19 стр. / Cvetkov K.Ju., Fedoseev V.E., Korovin V.M., Abazina E.S. Method for hidden data transfer in a video image. Patent for invention RUS 2608150, published 16.01.2017, 19 p. (in Russian).

[6]. Галушка В.В., Петренкова С.Б., Дзюба Я.В., Панченко В.А. Сетевая стеганография на основе ICMP-инкапсуляции. Инженерный вестник Дона, №4(51), 2018 г., стр. 107-118. / Galushka V.V., Petrenkova S.B., Dzjuba Ja.V., Panchenko V.A. Network steganography based on ICMP-encapsulation. Engineering journal of Don, №4(51), 2018, pp.107-118 (in Russian).

[7]. Стародубцев Ю.И., Закалкин П.В., Мартынюк И.А. Способ скрытного информационного обмена. Вопросы радиоэлектронники. Серия: Техника телевидения, №1, 2020 г., стр. 57-63. / Starodubcev Ju.I., Zakalkin P.V., Martynjuk I.A. Method of secret information exchange. Voprosy radiojelektronniki. Serija: tehnika televidenija, №1, 2020, pp. 57-63 (in Russian).

[8]. Иванов В.А., Снаров М.М., Двилянский А.А., Иванов И.В., Кирюхин Д.А., Крюков М.С., Ксенофонтов А.А., Щуров К.С. Способ встраивания информации в графический файл, сжатый фрактальным методом. Патент на изобретение RUS 2602670, опубл. 20.11.2016, 11 стр. / Ivanov V.A., Snarov M.M., Dviljanskij A.A., Ivanov I.V., Kirjuhin D.A., Krjukov M.S., Ksenofontov A.A., Shhurov K.S. A method of embedding information into a graphic file is compressed fractal method. Patent for invention RUS 2602670, published 20.11.2016, 11 p. (in Russian).

[9]. Короновский А.А., Москаленко О.И., Храмов А.Е. Способ скрытой передачи информации. Патент на изобретение RUS 2349044, опубл. 10.03.2009, 8 стр. / Koronovskij A.A., Moskalenko O.I., Khramov A.E. Secure information transmission method, Patent for invention RUS 2349044, published 10.03.2009, 8 p. (in Russian).

[10]. Москаленко О.И., Фролов Н.С., Короновский А.А., Храмов А.Е. Способ скрытой передачи информации. Патент на изобретение RUS 2509423, опубл. 10.03.2014, 11 стр. / Moskalenko O.I., Frolov N.S., Koronovskij A.A., Khramov A.E. Secure information transmission method. Patent for invention RUS 2349044, published 10.03.2009, 11 p. (in Russian).

[11]. Алексеев А.П., Макаров М.И. Способ скрытой передачи зашифрованной информации по множеству каналов связи. Патент на изобретение RUS 2462825, опубл. 27.09.2012, 39 стр. / Moskalenko O.I., Frolov N.S., Koronovskij A.A., Khramov A.E. Method of hidden transfer of coded information along multiple communication channel. Patent for invention RUS 2462825, published 27.09.2012, 39 p.

[12]. Котцов В.А., Котцов П.В. Способ скрытой передачи цифровой информации. Патент на изобретение RUS 2636690, опубл. 09.12.2016, 13 стр. / Kottsov V.A., Kottsov P.V. Method of hidden transferring digital information. Patent for invention RUS 2636690, published 09.12.2016, 13 p.

[13]. Бухарин В.В., Закалкин П.В., Кирьянов А.В., Стародубцев Ю.И. Способ маскирования передаваемой информации. Патент на изобретение RUS 2660641, опубл. 06.07.2018, 13 стр. / Buharin V.V., Zakalkin P.V., Kir'janov A.V., Starodubcev Yu.I. Method of masking transmitted information. Patent for invention RUS 2660641, published 06.07.2018, 13 p.

[14]. Закалкин П.В., Кирьянов А.В., Приходько А.В., Манзюк В.В. Программа маскирования передаваемой информации. Свидетельство о государственной регистрации программ для ЭВМ RU 2019618344. / Zakalkin P.V., Kir'janov A.V., Prihod'ko A.V., Manzjuk V.V. Program for masking transmitted information. Certificate of state registration of computer programs RU 2019618344.

[15]. Вентцель Е.С. Теория вероятностей. М., Академия, 2003 г., 576 стр. / Wentzel' E.S. Probability theory. M., Academy, 2003, 576 p. (in Russian).

Информация об авторах / Information about authors

Павел Владимирович ЗАКАЛКИН – кандидат технических наук, докторант. Научные интересы: информационная безопасность, безопасность компьютерных сетей.

Pavel Vladimirovich ZAKALKIN – Candidate of Technical Sciences, doctoral candidate. Research interests: information security, computer network security.

Сергей Александрович ИВАНОВ – кандидат технических наук, докторант. Сфера научных интересов: теория управления информационно-телекоммуникационными ресурсами, инфотелекоммуникационные системы и их подсистемы обеспечения.

Sergei Aleksandrovich IVANOV – Candidate of Technical Sciences, doctoral candidate. Research interests: theory of information and telecommunication resources management, infotelecommunication systems and their support subsystems.

Елена Валерьевна ВЕРШЕННИК – кандидат технических наук, преподаватель. Научные интересы: информационная безопасность, криптографические методы защиты информации.

Elena Valer’evna VERSHENNIK – Candidate of Technical Sciences, lecturer. Research interests: information security, cryptographic methods of information protection.

Александр Владимирович КИРЬЯНОВ – кандидат технических наук. Сфера научных интересов: безопасность компьютерных сетей, инфотелекоммуникационные системы и их подсистемы обеспечения.

Aleksandr Vladimirovich KIR’YANOV – Candidate of Technical Sciences. Research interests: security of computer networks, info-telecommunications systems and their supporting systems.

DOI: 10.15514/ISPRAS–2020–32(6)–10

Иерархическая рубрикация текстовых документов

Д.И. Сорокин, ORCID 0000-0002-6466-3714 <dmitrii.sorokin@phystech.edu>
А.С. Нужный, ORCID 0000-0003-3319-2523 <nuzhny@ibrae.ac.ru>
Е.А. Савельева, ORCID 0000-0002-6562-8750 <esav@ibrae.ac.ru>
Институт проблем безопасного развития атомной энергетики РАН,
115191, Россия, г. Москва, ул. Большая Тульская, д. 52

Аннотация. В работе представлены алгоритм и компьютерная программа иерархической рубрикации текстовой документации. Программа позволяет структурировать неупорядоченный корпус документов в виде иерархии рубрик и визуализировать результат в виде интерактивной карты. Для каждой рубрики автоматически определяются ключевые слова, по которым находятся документы, отнесенные к ней. Анализ построенной иерархии тем позволяет оценить минимальную и максимальную допустимую глубину иерархии, соответствующие минимальному и максимальному количеству различных тем, содержащихся в корпусе документов. Программа апробирована на коллекции документов по захоронению радиоактивных отходов. Результаты тестирования программы показывают хорошее качество построенной иерархии рубрик. Программа может быть использована для ознакомления с коллекцией документов и для тематического поиска.

Ключевые слова: рубрикация; иерархическая кластеризация; обработка естественного языка; машинное обучение

Для цитирования: Сорокин Д.И., Нужный А.С., Савельева Е.А. Иерархическая рубрикация текстовых документов. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 127-136. DOI: 10.15514/ISPRAS–2020–32(6)–10

Hierarchical Rubrication of Text Documents

D.I. Sorokin, ORCID 0000-0002-6466-3714 <dmitrii.sorokin@phystech.edu>
A.S. Nuzhny, ORCID 0000-0003-3319-2523<nuzhny@ibrae.ac.ru>
E.A. Saveleva, ORCID 0000-0002-6562-8750 <esav@ibrae.ac.ru>
Nuclear safety institute of the Russian Academy of Sciences,
52, Bolshaya Tulskaya st., Moscow, 115191, Russia

Abstract. Topic modeling is an important and widely used method in the analysis of a large collection of documents. It allows us to digest the content of documents by examination of the selected topics. It has drawbacks such as a need to specify the number of topics. The topics can become too local or too global, depending on that number. Also, it does not provide a relation between local and global topics. Here we present an algorithm and a computer program for the hierarchical rubrication of text documents. The program solves these problems by creating a hierarchy of automatically selected topics in which local topics are connected of the global topics. The program processes PDF documents split them into text segments, builds vector representations using word2vec model and stores them in a database. The vector embeddings are structured in the form of a hierarchy of automatically constructed categories. Each category is identified by automatically selected keywords. The result is visualized in an interactive map. Traversing the hierarchy of topics is done by zooming the map. An analysis of the constructed hierarchy of categories allows us to evaluate the minimum and maximum depth of the hierarchy corresponding to a minimum and a maximum number of different topics contained in the collection of documents. The program was tested on documents on deep nuclear waste disposal.

The results show good quality of the constructed hierarchy of topics and the program can be used for familiarization with the collection of documents and for thematic search.

Keywords: rubrication; hierarchical clustering; natural language processing; machine learning

For citation: Sorokin D.I., Nuzhny A.S., Saveleva E.A. Hierarchical rubrication of text documents. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 127-136 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–10

1. Введение

Рубрикация (отнесение документа к одной или нескольким категориям / рубрикам) является одним из наиболее распространенных методов систематизации не упорядоченной коллекции документов. Рубрикация позволяет определять набор тем содержащихся в документах и осуществлять быстрый поиск интересующей информации. В случае если темы рубрики заданы заранее, задачу рубрикации можно рассматривать как задачу классификации. Если же темы рубрик заранее не заданы, то рубрикацию можно рассматривать как задачу кластеризации. Отнесение текста к той или иной рубрике может быть выполнено на основании экспертной оценки, на основе правил или с использованием методов машинного обучения. Экспертная оценка большого корпуса текстов чаще всего является трудоемкой задачей. Рубрикация на основе правил требует подбора правил для каждой предметной области, так как одинаковые термины в разных предметных областях могут иметь различное значение. По этим причинам, в настоящее время большой интерес представляет разработка методов автоматической рубрикации с привлечением машинного обучения.

В данной работе рассматривается задача рубрикации в применении к документации по темам, связанным с захоронением радиоактивных отходов (РАО). При обосновании безопасности захоронения РАО учитывается множество факторов, связанных с геологией, гидрогеологией, геомеханникой, теплофизикой и радиохимией. Многие рассматриваемые факторы близки друг к другу по смыслу и в зависимости от цели информационного поиска можно выделять как более общие темы, связанные, например, с теплопереносом и радионуклидами, гидрогеологией и радиохимией так и более частные темы. Необходимость рассматривать большое количество факторов при обосновании безопасности пункта захоронения РАО приводит к значительному объему разнообразной документации, состоящей из книг, научных статей, отчетов и служебных документов. Для решения данной задачи требуется разработка методов и создание автоматизированных средств рубрикации и поиска документов.

Для рубрикации на основе машинного обучения каждый документ представляется в виде вектора в n-мерном пространстве. Широкое распространение получили подходы к векторизации текстов основанные на использовании нейронных сетей (автоэнкодеров), строящих отображение из исходного пространства в пространство той-же размерности через промежуточное низко размерное представление. Одни из лучших результатов достигнуты с помощью автоэнкодеров таких как word2vec [1], FastText [2], GloVe [3] и doc2vec [4]. Эти методы позволяют автоматически получать семантически нагруженные векторные представления слов или фрагментов документов, а полученные векторы затем могут быть использованы для контекстного поиска [5] или кластеризации документов. Также в настоящее время большого успеха достигли языковые модели, построенные с помощью нейронных сетей на архитектуре трансформеров. Языковая модель BERT [6] показала один из лучших результатов в задачах направленных на понимание текстов таких как ответы на вопросы. Векторное представление, получаемое в результате обучения такой модели, также может быть использовано для кластеризации текстов. Однако данная задача не требует построения языковой модели и сравнимое качество кластеризации может быть достигнуто значительно более простым и менее ресурсоемким подходом с использованием автоэнкодеров [7].

На практике часто возникает необходимость рубрикации набора (корпуса) документов, когда рубрики не заданы или заданы не точно. В этом случае использование алгоритмов кластеризации позволяет распределять документы в кластеры так что признаки входящих в один кластер документов близки по заданной мере. Дополнительной сложностью при кластеризации является выбор числа кластеров для корпуса текстов с заранее неизвестным количеством различных тем. Если число кластеров окажется слишком велико, то каждый кластер будет содержать только очень узко специализированные тексты. А при слишком маленьком количестве кластеров в отдельный кластер могут попадать настолько разнородные тексты, что для них сложно выделить общую тему. Между этими значениями числа кластеров может существовать несколько уровней кластеризации, которые достаточно хорошо описывают как глобальные, так и более локальные темы, встречающиеся в корпусе текстов. Для построения иерархии (дерева) вложенных кластеров могут использоваться методы иерархической кластеризации [8]. Эти методы итеративно строят иерархию кластеров, позволяющую отслеживать в зависимости от уровня более или менее глобальные темы. Анализ получившейся иерархии выполняется с помощью ключевых слов определяемых для каждого кластера. Ключевые слова соответствуют темам документов, вошедших в кластер и позволяют судить о качестве кластеризации.

В работе описывается программное средство, позволяющее построить взаимосвязанную иерархию рубрик по коллекции документов и визуализировать эту иерархию в виде карты. Подход основан на векторном представлении документов и методах иерархической кластеризации. В качестве расстояния между векторами документов используется L2 норма. Тестирование программы производилось на основе коллекции из 200 документов по теме глубинного захоронения радиоактивных отходов [9]. Каждый документ разбивался на смысловые фрагменты, что привело к корпусу из 150 тысяч фрагментов текстов. Для построенной иерархии тем выполнена оценка точности в зависимости от уровня иерархии.

2. Обзор методов иерархической кластеризации текстов

Методы иерархической кластеризации направлены на создание дерева вложенных кластеров. Для построения иерархической кластеризации выделяется два класса подходов агломеративные и дивизивные. При агломеративном подходе каждый документ изначально представляет собой отдельный кластер. Затем на каждом шаге выбираются два наиболее близких по заданной метрике кластера и объединяются в один. Новому кластеру ставится в соответствие вектор центра масс, входящих в него документов. При дивизивном подходе документы изначально представляют собой один кластер. На каждом шаге иерархической кластеризации один из кластеров разбивается на два. В отличии от методов с заранее заданным количеством кластеров таких как, например, k-means [10] иерархическая кластеризация не требует заранее определенных параметров и тем самым лучше подходит для реальных данных.

Применение иерархической кластеризации к текстовым документам рассматривалось в работах [11][13]. В работе [11] был предложен способ объединить агломеративную иерархическую кластеризацию с не иерархической кластеризацией. Было показано, что использование кластеризации алгоритмом k-means не уменьшает точность последующей агломеративной кластеризации, но позволяет ее значительно ускорить посредством уменьшения количества кластеризуемых данных. В работе [12] предложен метод агломеративной кластеризации в котором объединение в кластеры производится до тех пор, пока выполняется критерий «похожести». Данный подход выглядит похожим на подход применяемый в алгоритме dbscan [14]. В нем также вместо количества кластеров задается максимальное расстояние между двумя элементами, формирующими один кластер.

Важной задачей при анализе качества иерархической кластеризации является выделение ключевых слов для полученных кластеров. В работе [15] приведен подробный анализ

методов автоматического извлечения терминов. В зависимости от используемых предположений авторы разделяют их на следующие группы: методы, основанные на частотах слов; методы на основе контекстов вхождений; методы на основе тематических моделей; методы на основе внешних корпусов; методы на основе поисковых машин; методы на основе онтологий; методы на основе Википедии; методы на основе признаков. Для анализа иерархии кластеров нами был выбран метод называемый «частотность терминов – обратная частотность документов» (TF-IDF, Term frequency – inverse document frequency). TF-IDF метод вычисляет метрику как произведение нормированной частоты слова в документах данного кластера на инверсию частоты слова во всех документах

$$\mathrm{tf(t,D)} \cdot \ idf(t,D) = \frac{n_t}{\sum_k n_k} \cdot \log(\frac{|D|}{|\{d_i \in D \ | t \in d_i\}|}),$$

где n_i- число вхождений слова i в документы кластера, $|D|$ число документов в корпусе, $|\{d_i \in D \ | t \ \in d_i\}|$, число документов в корпусе, в которых встречается слово t.

В работах [16][13] предложены специальные методы для извлечения ключевых слов при иерархической кластеризации документов. Предложенные методы могут улучшить качество выделения ключевых слов, но так как ключевые слова в большинстве случаев не известны, то для оценки их работы требуется дополнительная экспертная оценка.

3. Иерархическая кластеризация с использованием карты Кохонена

В данной работе также, как и в [5] используется модель векторизации текста word2vec. Этот подход позволяет получить семантически нагруженные векторные представления (эмбеддинги) для слов из корпуса текстов. Подход word2vec заключается в обучении нейронной сети предсказывать центральное слово исходя из контекста. В результате обучения в скрытом слое формируется сжатое представление контекста, в котором данное слово употребляется в корпусе. Это скрытое представление затем используется в качестве эмбеддинга.

Для визуализации, а также для уменьшения размерности и сокращения количества векторов перед иерархической кластеризацией используется карта Кохонена [17]. Карта представляет собой двумерную гексагональную сетку. Отличие карты Кохонена от других методов снижения размерности таких как t-sne [18] заключается в том, что отображение, построенное при обучении карты, может быть использовано для преобразования новых данных без необходимости проводить обучение повторно. Это позволит добавлять в уже построенную иерархию новые документы.

Схема обработки документов представлена на рис. 1. Она состоит из двух основных блоков: предобработка документов и построение иерархии рубрик. Предобработка документов включает в себя считывание текста из исходного корпуса документов в формате PDF (tika.apache.org), разбиение текста документов на смысловые единицы – абзацы. Разбиение на абзацы необходимо так как большинство документов содержит в себе более одной темы. Текст каждого абзаца сохраняется в базу данных в формате sqlite (https://www.sqlite.org). Далее из текста удаляются стоп-слова – слова не несущие смысловой нагрузки такие как предлоги, союзы, междометия и производится нормализация слов – приведение слов к первому лицу, единственному числу, мужскому роду и нормальной форме глагола. Нормализация и удаление стоп-слов позволяет значительно уменьшить размер словаря и в конечном итоге увеличить качество рубрикации. На обработанных описанным выше способом фрагментах текстов обучается модель word2vec. После обучения модели для каждого абзаца вычисляется вектор центра масс, входящих в него векторов слов, полученных из word2vec и также сохраняется в базу данных. Данной подход соответствует приближению «мешка слов» в котором порядок слов во фрагменте текста не имеет значения, имеет значение лишь их наличие.

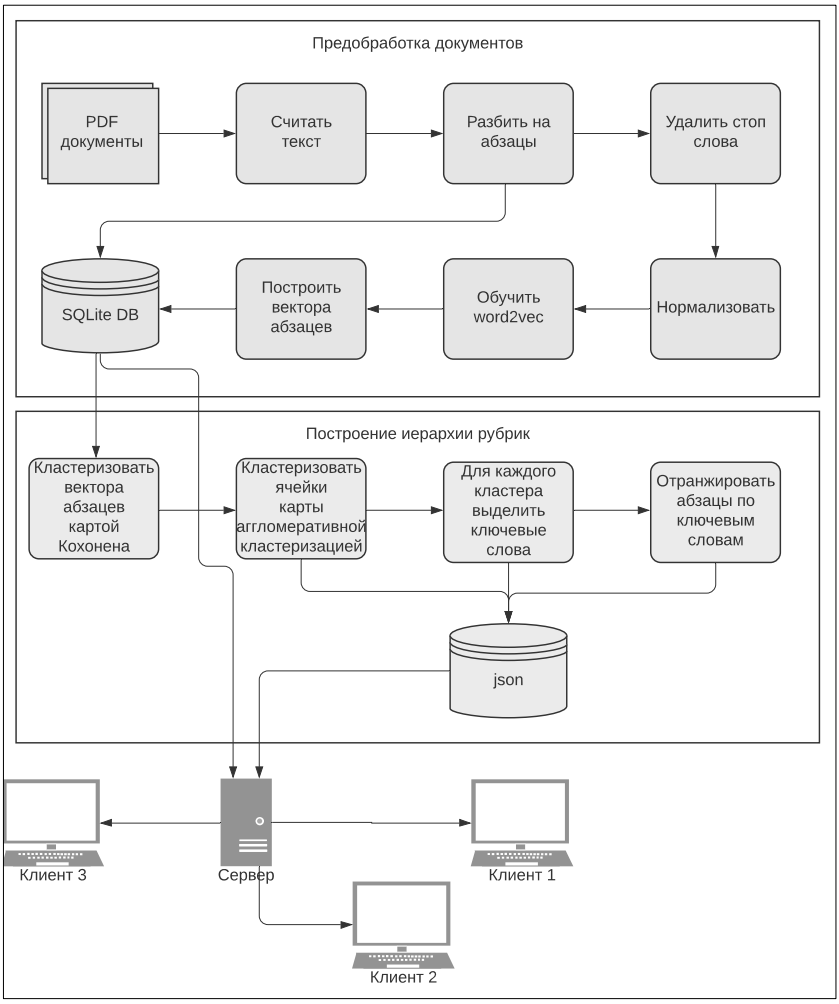

Рис. 1. Схема обработки документов
Fig. 1. Document preprocessing diagram

Далее осуществляется построение иерархии рубрик. На основе векторов абзацев из базы данных строится карта Кохонена. Ячейки карты Кохонена затем кластеризуются с помощью агломеративной кластеризации (в реализации scikit-learn). Для каждого кластера выделяется набор ключевых слов на основе метрики $M = \frac{n_t^2}{\sum_k n_k}$. Для последующего поиска наиболее релевантных к теме кластера документов, фрагменты текстов, отнесенные к каждому кластеру, ранжируются на основании ключевых слов кластера с помощью алгоритма okapi-bm25 [19]. Результат кластеризации, ключевые слова кластеров и ранжированные идентификаторы документов сохраняются в файл в формате json (https://www.json.org).

База данных и json-файл затем используется в клиент-серверном приложении.

4. Анализ результатов

В случае построения иерархии рубрик нужно определять минимальную и максимальную глубину иерархии такую, чтобы выделенные рубрики не были слишком частными или же слишком общими. Для решения данной проблемы оценим качество выделения ключевых слов в зависимости от количества рубрик. Зависимость качества выделения ключевых слов

от шага иерархической кластеризации представлена на рис. 2. По оси абсцисс отложен шаг иерархической кластеризации: на нулевом шаге каждый документ представляет собой отдельный кластер, на последнем шаге все документы объединены в один кластер. По оси абсцисс справа отложено расстояние между двумя кластерами, объединенными в один на текущем шаге кластеризации. Видно, что это расстояние в начале растет слабо так как, объединяются близкие кластеры, а в конце растет значительно быстрее так как начинают объединяться сильно удаленные друг от друга кластеры.

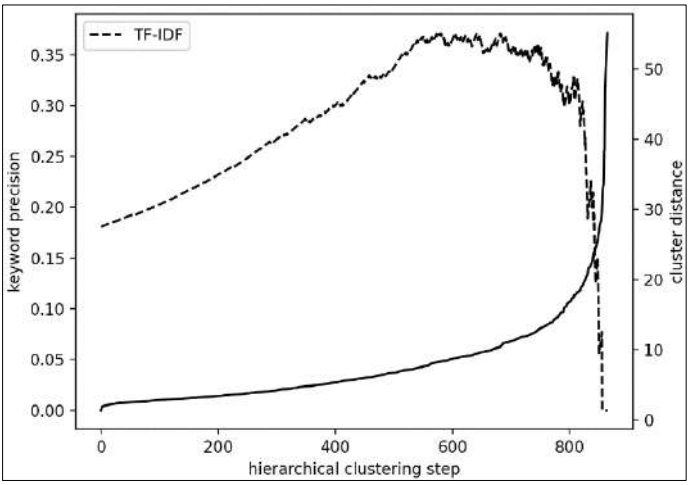

Рис. 2. Оценка качества иерархической кластеризации в зависимости от шага кластеризации
Fig. 2. Hierarchical clustering quality as a function of hierarchical clustering step

На оси ординат слева отмечено качество выделения ключевых слов – число правильно выделенных ключевых слов деленное на общее число выделенных ключевых слов в кластере. Для оценки качества необходимо знать «правильные» ключевые слова. В качества «правильных» ключевых слов рассматривались ключевые слова, выбранные экспертами [20]. Каждому фрагменту текста ставилось в соответствие подмножество (не более 10) ключевых слов, отсортированных по частотам появления во фрагменте текста.

Из рис. 2 видно, что при слишком большом размере кластера на последних шагах иерархической кластеризации сильно растет расстояние между кластерами и одновременно с этим сильно падает точность выделения ключевых слов. Также при слишком маленьком размере кластера получается достаточно низкая точность выделения ключевых слов. Между этими двумя областями находится зона, в которой точность выделения ключевых слов достигает максимума и некоторое время остается на этом уровне. Эта область соответствует иерархии тем, которые лучшим образом описывают данную коллекцию документов. Ограничение глубины иерархической кластеризации размерами близкими к этой области позволяет значительно улучшить качество получаемой иерархии.

5. Интерфейс

Предложенный алгоритм реализован в виде клиент-серверного приложения, написанного на flask (https://flask.palletsprojects.com). Клиентское окно программы состоит из двух частей (Рис. 3). Слева отображается гексагональная карта Кохонена в которой различные кластеры обозначены различным цветом. Приближение карты позволяет переходить между уровнями иерархической кластеризации. При наведении курсора на кластер, кластер подсвечивается и над курсором отображаются ключевые слова. При нажатии на кластер справа отображаются документы отнесенные к этому кластеру. Программа позволяет ознакомиться с иерархией тем содержащихся в документах и найти документы, отнесенные к выбранной рубрике.

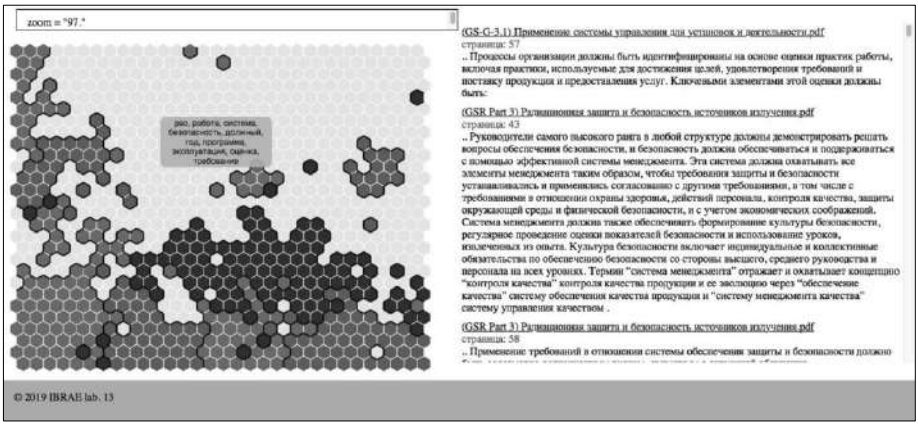

Рис. 3. Окно программы иерархической рубрикации
Fig. 3. Hierarchical rubrication program window

Рассмотрим пример использования программы. Допустим пользователь ознакомился с рубриками верхнего уровня и его интересует рубрика «РАО, работа, система, безопасность, должный, год, программа, эксплуатация, оценка, требование». При приближении карты он видит, что эта рубрика в свою очередь состоит из двух рубрик:

- (1a) «РАО, отход, здание, контейнер, радиоактивный отход, оборудование, помещение, система, контроль, захоронение»;
- (1b) «Программа, безопасность, оценка, работа, год, должный, модель, эксплуатация, требование, решение».

При последующих приближениях карты каждая из рубрик в свою очередь перейдет в составляющие подрубрики. Рубрика (1a) состоит из рубрик:

- (2aa) «Здание, помещение, система, контейнер, контроль, устройство, фильтр, машина, средство»;
- (2ab) «РАО, отход, радиоактивный отход, обращение, захоронение, хранение, переработка, упаковка, реактор, ТРО».

Рубрика (1b) состоит в свою очередь из рубрик:

- (2ba) «Программа, безопасность, требование, организация, управление, должный, МАГАТЭ, атомный энергия, документ» и
- (2bb) «Модель, оценка, облучение, доза, метод, условие, параметр, работа».

Видно, что рубрики верхнего уровня являются более общими. По мере увеличения глубины тема становится более конкретной и в тоже время прослеживается связь с родительской рубрикой. Когда пользователь определится с интересующей рубрикой и кликнет по ней справа откроется список фрагментов текстов, ранжированный, по ключевым словам, выбранной рубрики.

Код приложения выложен на github (https://github.com/dmitrySorokin/cluster_search). Там можно увидеть пример взаимодействия пользователя с интерфейсом программы. Ключевым отличием разработанной программы является использование карты Кохонена, которая позволяет не только визуализировать кластеры документов, но также отражает соотношения между ними: близкие на карте кластеры состоят из близких по темам документов; размер кластера отражает количество документов, отнесённых к нему. Это является ключевым отличием предложенного подхода по сравнению с программой Carrot2 (https://search.carrot2.org/#/search/web/hierarchical%20clustering/pie-chart), решающей похожую задачу с помощью круговых диаграмм. Размер кластеров позволяет оценивать

количество документов, отнесенных к заданной теме и находить недостаточно представленные в коллекции документов темы.

5. Заключение

В данной работе представлен автоматический способ рубрикации и визуализации не структурированного корпуса текстов. Предложенный алгоритм применен к коллекции документов по теме глубинного захоронения радиоактивных отходов. Полученный результат позволяет визуализировать эту коллекцию документов в виде карты и производить поиск по автоматически выделенным темам. Анализ полученных результатов показал хорошее соответствие автоматически построенной иерархии рубрик, рубрикам, заданным экспертными ключевыми словами. Также была выделена область иерархической кластеризации, которая лучшим образом соответствует иерархии тем в коллекции документов. Данная программа может быть использована для первичного ознакомления с содержанием коллекции документов и поиска по рубрикам. В дальнейшем планируется перейти от описания темы с помощью ключевых слов к методам суммаризации текстов.

Список литературы / References

[1]. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. In Proc. of the International Conference on Learning Representations, Workshop Track, 2013, 12 p.

[2]. Bojanowskij P., Grave E., Joulin A., and Mikolov T. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, vol. 5, 2017, pp. 135-146.

[3]. Pennington J., Socher R., Manning C. GloVe: Global Vectors for Word Representation. In Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532-1543.

[4]. Le Q., Mikolov T. Distributed representations of sentences and documents In Proc. of the 31st International Conference on Machine Learning, 2014, pp. 1188-1196.

[5]. Нужный А.С., Сорокин Д.И. Создание программы интеллектуального анализа текстовой документации по вопросам захоронения РАО. Труды МФТИ, том 12, № 1(45), 2020 г., стр. 104-111 / Nuzhny A.S., Sorokin D.I. Development of a text-mining program for analysis of documentation on the disposal of radioactive wasteproblem. Proceedings of MIPT, vol. 12, № 1(45), 2020, pp. 104-111 (in Russian).

[6]. Devlin J., Chang M.-W., Lee K., Toutanova K. Bert: Pre-training of Deep Bidirectional Transformers for Language understanding. In Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, 2019, pp. 4171-4186.

[7]. Sia S., Dalmia A., Mielke S.J. Tired of Topic Models? Clusters of Pretrained Word Embeddings Make for Fast and Good Topics too! In Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 1728–1736.

[8]. Mullner D. Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378v1, 2011, 29 p.

[9]. Свительман В.С., Савельева Е.А., Бутов Р.А., Линге Ин.И., Дорофеев А.Н., Тихоновский В.Л. Информационно-аналитическая платформа программы исследований по обоснованию долговременной безопасности российского ПГЗРО. Радиоактивные отходы, № 2 (3), 2018 г., стр. 79-87 / Svitelman V.S., Dorofeev A.N., Saveleva E.A., Butov R.A., Linge I.I., Tikhonovsky V.L. Informational and Software Environment of the Russian Deep Geological Repository Research Program. Radioactive Waste, № 2 (3), 2018, pp. 79–87 (In Russian).

[10]. Jin X., Han J. K-Means Clustering. In Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining, Springer, 2011.

[11]. Bouguettaya A., Yu Q., Liu X., Zhou X., Song A. Efficient agglomerative hierarchical clustering. Expert Systems with Applications, vol. 42, issue 5, 2015, pp. 2785-2797.

[12]. Peng T., Liu L. A novel incremental conceptual hierarchical text clustering method using CFu-tree. Applied Soft Computing, vol. 27, 2015, pp. 268-278.

[13]. Nagarajan R., Nair S.A.H., Puviarasan N., Aruna P. Document clustering using agglomerative hierarchical clustering approach (AHDC) and proposed TSG keywords extraction method. IJRET: International Journal of Research in Engineering and Technology, vol. 05, issue 18, 2016, pp. 118-124.

[14]. Ester M., Kriegel H., Sander J., and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proc. of the 2nd ACM International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226-231.
[15]. Астраханцев Н.А., Федоренко Д.Г., Турдаков Д.Ю. Методы автоматического извлечения терминов из коллекции текстов предметной области. Программирование, том 41, № 6, 2015 г., стр. 33-52 / Astrakhantsev N.A., Fedorenko D.G., Turdakov D.Yu. Methods for automatic term recognition in domain-specific text collections: A survey. Programming and Computer Software, vol. 41, № 6, 2015, pp. 336-349.
[16]. Peganova I., Rebrova A., and Nedumov Y. Labelling Hierarchical Clusters of Scientific Articles. In Proc. of the 2019 Ivannikov Memorial Workshop (IVMEM), 2019, pp. 26-32.
[17]. Kohonen T. Self–Organizing Maps. Springer, 1997, 426 p.
[18]. van der Maaten L., Hinton G. Visualizing Data using t-SNE. Journal of Machine Learning Research, vol. 9, 2008, pp. 2579-2605.
[19]. Robertson S.E., Walker S., Beaulieu M. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive. In Proc. of the Seventh Text Retrieval Conference, 1998, pp. 253-264.
[20]. Рукавичникова А.А., Валетов Д.К., Бутов Р.А., Свительман В.С. Средства тематической кластеризации документов для систематизации библиографической информации по вопросам ПГЗРО. Сборник трудов XIX научной школы молодых ученых ИБРАЭ РАН, 2018 г., стр. 145-148 / Rukavichnikova A.A., Valetov D.K., Butov R.A., Svitelman V.S. Tools for thematic clustering of documents for systematization of bibliographic information on the issues of PGWDF. In Proc. of the XIX Scientific School of Young Scientists IBRAE RAS, 2018, pp. 145-148 (in Russian).

Информация об авторах / Information about authors

Дмитрий Игоревич СОРОКИН – инженер. Научные интересы: обработка естественного языка, глубокое обучение, обучение с подкреплением.

Dmitry Igorevich SOROKIN – engineer. Research interests: natural language processing, deep learning, reinforcement learning.

Антон Сергеевич НУЖНЫЙ – кандидат физико-математических наук, старший научный сотрудник. Научные интересы: теория машинного обучения, некорректные задачи, обработка естественного языка, распознавание образов.

Anton Sergeevich NUZHNY – Ph.D. in Physical and Mathematical Sciences, senior researcher. Research interests: theory of machine learning, ill-posed problems, natural language processing, pattern recognition.

Елена Александровна САВЕЛЬЕВА – кандидат физико-математических наук, заведующая лабораторией геостатистического моделирования. Научные интересы: статистические методы анализа данных, чувствительность модели к ее параметрам, неопределенность при моделировании.

Elena Alexandrovna SAVELEVA – Ph.D. in Physical and Mathematical Sciences, head of geostatistical laboratory. Research interests: statistical methods of data analysis, sensitivity analysis, uncertainty in modeling.

DOI: 10.15514/ISPRAS–2020–32(6)–11

Обзор методов классификации сетевого трафика с использованием машинного обучения

[1,2] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <thorin@ispras.ru>*
[1] *М.К. Иконникова, ORCID: 0000-0003-1530-5133 <mikonnikova@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный исследовательский университет «Высшая школа экономики», Россия, 101000, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В статье рассматривается задача классификации сетевого трафика с использованием методов машинного обучения. Приводятся различные постановки задачи, описываются ограничения использовавшихся ранее методов и причины использования машинного обучения в данной области. Рассматриваются различные алгоритмы машинного обучения, которые могут использоваться для решения задачи, указываются их преимущества и недостатки. Исследуется вопрос отбора признаков для классификации и проблема получения данных для обучения, основные компромиссы в этом вопросе. Перечисляются часто используемые наборы данных и их характеристики. Завершается обзор описанием актуальных проблем в данной области: обучение и сравнение моделей, защита данных пользователей, изменчивость трафика.

Ключевые слова: анализ сетевого трафика; классификация сетевого трафика; машинное обучение

Для цитирования: Гетьман А.И., Иконникова М.К. Обзор методов классификации сетевого трафика с использованием машинного обучения. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 137-154. DOI: 10.15514/ISPRAS–2020–32(6)–11

A survey of Network Traffic Classification Methods Using Machine Learning

[1,2] *A.I. Getman, ORCID: 0000-0002-6562-9008 <thorin@ispras.ru>*
[1] *M.K. Ikonnikova, ORCID: 0000-0003-1530-5133 <mikonnikova@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *National Research University Higher School of Economics, 20, Myasnitskaya st., Moscow, 101000, Russia*

Abstract. This survey is dedicated to the task of network traffic classification, particularly to the use of machine learning algorithms in this task. The survey begins with the description of the task, its variations and possible uses in real-world problems. It then proceeds to the description of the methods used historically to solve this task, their limitations and evolution of traffic making machine learning the main way to solve the problem. Then the most popular machine learning algorithms used in this task are described, with the examples of research papers, providing the insight into their advantages and disadvantages in relation to this field. The task of feature selection is discussed, followed by the more global problem of acquiring the suitable dataset to use in the research; some examples of such popular datasets and their descriptions are provided. The paper concludes with the outline of the current problems in this research area to be solved.

Keywords: network traffic analysis; network traffic classification; machine learning

For citation: Getman A.I., Ikonnikova M.K. A survey of network traffic classification methods using machine learning. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 137-154 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–11

1. Введение

Классификация трафика является необходимой в наше время, так как полученные результаты могут применяться в различных приложениях, важных как для администрирования сети, так и для конечного пользователя [1, 2].

С точки зрения провайдера определение протоколов/приложений/типов приложений по потокам данных в сети может использоваться для:

- контроля сети и трафика в ней (например, для блокировки отдельных протоколов, таких как BitTorrent),
- обеспечения высокого качества обслуживания клиентов посредством эффективного выделения наиболее приоритетных потоков и регулирования скорости передачи отдельных пакетов,
- регулирования цен на услуги,
- планирования размещения и использования ресурсов,
- оптимизации предоставляемых сервисов и алгоритмов маршрутизации (например, для изменения приоритетов передачи различных типов данных в случае высокой загруженности сети).

Оценка текущего использования сети пользователями может давать понимание об оптимальном устройстве новых сетей с учётом понимания предпочтений и принципов работы интернет-пользователей и интернет-сервисов, так как появляется возможность получать подробную статистику по всем сервисам.

Так как потребности пользователей относительно использования сети постоянно меняются, необходимо их знать и модифицировать Сеть в соответствии с актуальными запросами. Для этого нужно как уметь моделировать устройство сети на текущий момент времени, так и понимать направление движения её развития и изменения. Например, на сегодняшний день видна тенденция отказа от превалирующего ранее принципа асимметрии устройства сети в том смысле, что клиенты загружают намного больше информации, чем отправляют её в Сеть. Появление P2P-приложений, VoIP, видеозвонков, потоковой передачи мультимедиа и прочих новшеств должно вызвать у интернет-провайдеров соответствующие ответные действия по переустройству сети под новые запросы клиентов. Кроме того, в настоящее время увеличивается количество так называемых «умных устройств», которые должны в будущем составить Интернет вещей: он также поставит перед интернет-провайдерами ряд задач для обеспечения максимальной эффективности своей работы.

Отдельно следует упомянуть мобильные приложений, чья доля в интернет-трафике неуклонно растёт. Использование смартфонов и мобильных приложений можно считать более персонализированным, поэтому получение данных о такого рода трафике позволяет эффективно составлять сетевой портрет пользователя. Определение интересов пользователей может служить целям маркетинга, позволяя проводить лучше таргетированные рекламные кампании.

С точки зрения безопасности информационных систем, классификация интернет потоков может использоваться как важный признак при выявлении кибер-атак, аномалий в работе Сети, неправомерных или необычных действий пользователя и прочих нарушений, что способно повысить общую безопасность Сети.

Методы, применяемые для классификации интернет-трафика, меняются вместе с глобальными изменениями в устройстве трафика. Внедрение новых технологий начинает негативно влиять на качество работы ранее применимых способов, что приводит к

необходимости создания и развития новых походов. К таким глобальным изменениям, влияющим на решение задачи классификации трафика можно отнести:

- отказ от использования утверждённого списка портов в зависимости от протокола/приложения (намеренный или в связи с устареванием данного списка);
- обфускация протоколов с целью замаскировать те из них, которые блокируются/подавляются провайдером;
- всё более широкое распространение шифрования трафика, не позволяющее использовать для классификации содержимое полезной нагрузки пакета;
- постоянное появление новых протоколов и приложений и т.д.

По приведённым выше причинам, задачу классификации интернет трафика на сегодняшний день нельзя считать решённой, и исследовательские группы продолжают предлагать всё новые решения, позволяющие показывать эффективные результаты в условиях меняющейся реальности.

2. Эволюция методов классификации трафика

Методы классификации сетевого трафика с годами развиваются и модифицируются. Это связано в первую очередь с предъявляемыми сетью требованиями и ограничениями. Изменение устройства сетевого трафика и особенностей его передачи приводит к тому, что старые методы классификации становятся малоэффективными или просто непригодными. С другой стороны, развитие методов классификации и оборудования, на котором может работать система позволяет использовать больше признаков и более развитые способы их применения для принятия решения.

К важным характеристикам методов классификации сетевого трафика относятся:

- *детализация:* с каким уровнем точности система производит классификацию: семейство протоколов/класс приложений или конкретные протоколы, конкретные приложения.
- *скорость реакции:* способна ли система производить классификацию быстро (после нескольких пакетов), что подходит для анализа в реальном времени или для классификации нужны данные о потоке полностью.
- *вычислительная стоимость:* сложность вычислений и затраты по использованию памяти для классификации пакета или потока.

2.1 Классификация по номерам портов

Первые системы классификации трафика основывались на извлечении из пакетов номеров портов и сопоставлении их со списком IANA (Internet Assigned Numbers Authority, «Администрация адресного пространства Интернет»). IANA выделяет и регистрирует номера портов, используемые для конкретных специфических целей, например, под протокол HTTP выделен порт 80. Информацию о протоколе можно уже использовать для примерного определения типа деятельности пользователя. Этот метод классификации работает очень быстро и не требует хранения данных о потоке, вычислительно прост. Это позволяет, например, удобно использовать его в межсетевых экранах для фильтрации трафика. Однако, он обладает рядом существенных недостатков, которые по мере эволюции устройства Сети негативно влияет на результаты его работы.

Номер порта определён не для всех протоколов. В списке IANA уже содержатся категории «известно несколько применений наряду с зарегистрированным» и «порт не зарегистрирован IANA». Некоторые протоколы выбирают порты для обмена данными в ходе своей работы случайным образом (как FTP). Вдобавок, некоторые протоколы могут использовать известные номера портов других протоколов, чтобы замаскироваться под них, если другой протокол является более предпочтительным с точки зрения интернет-провайдера. Например,

протокол BitTorrent может таким образом маскироваться под HTTP, чтобы избегать блокировок или ограничений на скорость передачи данных. Появляющиеся в последнее время протоколы также могут не успевать получить зарезервированный за собой порт.

Этот метод хорошо подходит для определения протоколов, однако не способен хорошо различать приложения. Например, браузинг веб-страниц, VoIP и просмотр видео – все будут использовать 80 (HTTP) или 443 (HTTPS) порт для своей работы. Но у этих приложений абсолютно разные сценарии использования, поэтому на практике нам хотелось бы их различать.

Широкое распространение технологий туннелирования, инкапсулирующих протоколы, шифрование на уровне IP, использование NAT (Network Address Translation, преобразование сетевых адресов) и NAPT (Network Address and Port Translation, трансляция сетевых адресов и портов) – всё это влияет на применимость данного метода. Поэтому точность систем, основанных на определении номеров портов, невысока (по разным оценкам, от 30 до 70%) и продолжает ухудшаться. В настоящее время этот признак может служить лишь одним из многих, выступая как источник дополнительной информации при принятии решения, основанного на других критериях.

2.2 Глубокий анализ пакетов

Следующим шагом развития классификаторов интернет трафика стало использование технологии DPI (Deep Packet Inspection, глубокий анализ пакетов). Фильтрация сетевых пакетов в этом случае проводится по их полному содержимому, то есть проводится анализ не только заголовков, но и всего трафика на уровнях модели OSI со второго и выше. Этот метод показывает высокую точность работы, а полученная с его помощью разметка зачастую принимается как эталонная для данных с неизвестными классами. Для классификации с помощью DPI создаётся библиотека сигнатур и шаблонов пакетов, и для каждого пакета производится поиск соответствий в этой библиотеке.

Было замечено, что некоторые проприетарные протоколы передают информацию на уровне битов, что привело к созданию инструментов, работающих и на этом уровне. Генерируемые маски содержат значения 0, 1 и *[3] или вероятность единицы в данном бите [4].

При всех своих достоинствах метод DPI сталкивается с существенными проблемами в своей работе. Среди главных – невозможность работы с зашифрованным трафиком, доля которого в Интернете растёт с каждым годом, и высокие требования к ресурсам. Для хранения данных пакетов и библиотеки сигнатур требуется достаточно большой объём памяти, а при росте количества известных классов растёт размер этой библиотеки и, соответственно, время на поиск соответствий в ней. Поэтому, этот метод плохо подходит для работы в высокоскоростных сетях в режиме реального времени. Кроме того, определённую сложность представляет создание и поддержание в актуальном состоянии библиотеки сигнатур при всё увеличивающемся количестве протоколов и приложений в Сети.

Отдельно стоит вопрос защиты приватности пользователей Сети – проблема, которая актуальна для всех систем, использующих в своей работе полезную нагрузку пакетов. Законодательную сторону этого аспекта нужно учитывать при создании, обучении и работе систем глубокого анализа пакетов. Некоторые методы пытаются ограничить количество используемых данных пакета, например, первыми 40 битами [4], но полностью проблему это не решает.

2.3 Стохастический анализ пакетов

Стохастический анализ пакетов (SPI, Stochastic packet inspection) для классификации пакетов изучает статистические свойств их содержимого. Например, в [5] используется критерий Хи-квадрат Пирсона для изучения случайности распределения первых байтов полезной нагрузки

пакета. Таким образом строится модель синтаксиса протокола, используемого приложением. В [6] потоки определяются как зашифрованные или незашифрованные на основании энтропии первого пакета. В [7] вычисление энтропии первых байтов полезной нагрузки идентифицирует тип содержимого как текст, бинарный файл или шифрованный файл, что позволяет приоритизировать передачу некоторых файлов. Однако, такую классификацию сложно назвать точной или детализированной, так как для одного и того же приложения возможно использование всех видов содержимого. Кроме того, хотя стохастический анализ и использует более простые операции, чем глубокий анализ пакетов, он всё равно использует большой объём памяти для анализа. В связи с этим, данный метод не получил широкого распространения.

2.4 Использование машинного обучения для классификации трафика

Тенденции изменения сетевого трафика, широкое распространение шифрования, рост скорости передачи данных, а соответственно и необходимой скорости их обработки, постоянное появление новых классов трафика - всё это потребовало появления новых способов его классификации. Для этого были предложены методы машинного обучения, которые позволяют во многом упростить работу с созданием наборов различающих характеристик классов, автоматизируя этот процесс на основе анализа большого количества примеров этих классов (собрать который значительно проще, чем проанализировать вручную). Кроме того, многие из предложенных методов работают с общими признаками потоков, а не с полезной нагрузкой пакетов, что решает проблемы, связанные с шифрованием и с защитой данных пользователей. Это же даёт преимущество в скорости классификации и уменьшает необходимый для принятия решения объём памяти.

Далее будут подробно рассмотрены именно методы машинного обучения для классификации сетевого трафика: типы классификации, используемые модели и признаки, а также наборы данных, на которых производится обучение и тестирование моделей.

3. Типы классификации

Трафик в сети можно классифицировать отдельными пакетами, и методы классификации на основе портов и DPI способны решать эту задачу, однако сейчас в большинстве работ классификация производится для потоков. Здесь и далее, поток – это пятёрка значений:

<IP-адрес источника, IP-адрес получателя, порт источника, порт получателя, тип транспортного протокола>.

Существуют подходы, в которых запросы отправителя и ответы получателя интерпретируются как два разных встречных потока, однако более частым решением является объединение этих потоков в единый двунаправленный. Поскольку интернет-поток как правило подразумевает один законченный сеанс взаимодействия между отправителем и получателем (клиентом и сервером), проблем с классификацией всего потока как единого целого в один класс обычно не возникает.

Так как потоки различаются по своей продолжительности и количеству передаваемых данных, среди них иногда особо выделяют самые маленькие («мышиные», mice) потоки и самые большие («слоновьи», elephant). Из-за существенного отличия в объёме этих потоков результаты классификации иногда проверяются отдельно по доле правильно классифицированных потоков и по доле классифицированных байтов. Большое значение имеет также длина потока. Потоки или обрывки потоков, состоящие из малого количества пакетов, могут не нести достаточного количества информации для определения класса, поэтому нуждаются в специальном подходе или игнорируются.

Классификация трафика может проводиться в онлайн режиме, то есть в режиме реального времени, или офлайн, постфактум. Режим классификации определяется решаемой задачей.

Так, например, сбор статистики использования сети для глобального перераспределения ресурсов, получение информации об активности пользователя для выставления ему счетов за интернет-услуги и перерасчёт этих цен - всё это не требует срочного ответа и может обрабатываться в свободном режиме, с любым количеством доступных данных и ресурсов. Другие же задачи, такие как обеспечение качества сервиса для пользователя, выявление атак и угроз, оперативное перераспределение ресурсов, требуют как можно более быстрой реакции. В этих случаях классификатор не имеет возможности ждать окончания потока и оперировать полной информацией о нём, а вынужден ограничиваться лишь частью информации, например, лишь первыми N пакетами потока. Также накладываются ограничения на используемую модель классификатора, так как она должна достаточно экономно расходовать ресурсы системы и максимально оперативно принимать решение о потоке. Кроме того, в этих случаях решение обычно принимается параллельно для нескольких/многих потоков данных сразу, поэтому ограничения накладываются и на количество оперативной памяти, выделяемой для обработки потока.

Выбор набора классов для проведения классификации трафика зависит от решаемой задачи. В качестве примеров можно привести следующее.

1) Классификация по протоколам прикладного уровня (HTTP, SMTP, SSH и т.д.) [8, 9]. Обычно выбираются именно протоколы прикладного уровня, так как такая классификация является наиболее практически ценной.
2) Классификация по приложениям, генерирующим интернет-трафик (Skype, Torrent, браузер и т.д.) [10, 11]. Это определяет активность пользователя и позволяет строить его профиль, ограничивать деятельность конкретных приложений, решать маркетинговые задачи.
3) Классификацию по типам действий пользователя (интернет-браузинг, скачивание файлов, просмотр видео и т.п.) [12, 13]. В данном случае определяется не конкретное используемое приложение, а вид деятельности пользователя. В некотором смысле это обобщение предыдущего типа классификации.

Существуют и другие [14, 15] подходы к определению набора классов для классификации интернет-трафика, но приведённые три являются наиболее популярным, и именно они чаще всего исследуются и описываются в научных статьях.

Отдельно здесь следует упомянуть классификацию мобильного трафика [16] и классификацию трафика устройств интернета вещей [14], которые обладают особенностями, выделяющими их в отдельные задачи. Это и большая популярность новых и/или специализированных протоколов, и другие сценарии работы приложений, и другие масштабы распространения.

4. Методы машинного обучения, используемые для классификации трафика

Задача классификации сетевого трафика, как и другие задачи классификации, обычно рассматривается как задача обучения с учителем, поэтому при её решении используются соответствующие методы машинного обучения. Среди них можно выделить:

- наивный байесовский классификатор;
- метод опорных векторов;
- метод k-ближайших соседей;
- деревья принятия решений (с разными алгоритмами построения дерева: CART, C4.5, C5.0);
- методы бэггинга (случайный лес);
- методы бустинга (Adaboost, XGBoost);

- разные виды нейронных сетей: CNN, CNN+RNN, CNN+LSTM, SAE.

Рассмотрим примеры и результаты применения вышеупомянутых методов в исследованиях по теме классификации сетевого трафика.

4.1 Наивный байесовский классификатор

Наивный байесовский классификатор – простой вероятностный классификатор, основанный на применении Теоремы Байеса со строгими (наивными) предположениями о независимости. По теореме Байеса,

$$P(A|B) = \frac{P(B|A)*P(A)}{P(B)},$$

где A – класс, B – признак. Для предсказания неизвестного класса вычисляется его апостериорная вероятность. Все признаки считаются независимыми, вероятности напрямую вычисляются из обучающих данных (дискретные значения) или оцениваются через нормальное распределение.

Наивный байесовский классификатор является одним из самых простых методов машинного обучения, к его достоинствам можно отнести высокую скорость обучения и работы, однако он зачастую показывает результаты хуже, чем другие методы, поэтому его применение на практике ограничено. Можно упомянуть пример статьи [11], в котором он проигрывает в сравнении с другими рассмотренными классификаторами при классификации трафика, но показывает хорошие результаты как способ обобщения предсказаний разных типов классификаторов (см. подразд. 4.5, бэггинг).

4.2 Метод опорных векторов

В методе опорных векторов (SVM, support vector machine) каждый объект данных представляет из себя точку в p-мерном пространстве (где p - количество признаков), и алгоритм пытается построить гиперплоскость размерности (p−1), максимально эффективно разделяющую точки, относящиеся к разным классам. Таким образом возможна классификация на два класса, для проведения многоклассовой классификации на N классов посредством метода опорных векторов могут использоваться техники:

- попарного сравнения: строится N*(N-1)/2 классификаторов, каждый из которых учится различать между собой два класса;
- один против всех: строится N классификаторов, каждый из которых учится отличать один класс от всех остальных.

К достоинствам метода опорных векторов можно отнести его эффективность в многомерных пространствах признаков. Однако, этот метод достаточно затратен по памяти и вычислительной сложности и может легко переобучаться, то есть подстраиваться только под параметры конкретного множества примеров, на которых он обучался. В то же время его результаты для задачи классификации сетевого трафика хуже, чем у большинства других классификаторов [11], поэтому в настоящее время он редко используется с этой целью.

4.3 Метод k-ближайших соседей

Метод k-ближайших соседей присваивает каждому классифицируемому примеру то значение класса, которое наиболее распространено среди его k ближайших согласно выбранной функции расстояния соседей, классы которых известны. В качестве функции расстояния часто выбирается евклидова метрика. Такой подход не даёт высокую точность сам по себе, но может применяться в сочетании с кластеризацией, как в [17] для улучшения результатов выделения из трафика отдельных неизвестных классов.

4.4 Дерево принятия решений

Дерево принятия решений (decision tree) представляет собой бинарное дерево, в вершинах которого записаны атрибуты (признаки), по которым различаются различные ситуации, на рёбрах – значения этих атрибутов, а в листьях – значения целевой функции. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение.

Для построения дерева могут применяться разные алгоритмы, в работах по классификации сетевого трафика можно найти примеры использования:

- C4.5 [8, 18-20] – улучшенная версия одного из базовых алгоритмов построения деревьев ID3 (Iterative Dichotomiser 3), которая может использовать как дискретные, так и непрерывные признаки, позволяет задавать веса для признаков, и производит прореживание (pruning) для построенных деревьев с целью их оптимизации. Для выбора признака для разбиения используется информационный выигрыш, основанный на энтропии.
- C5.0 [10, 21] – оптимизация алгоритма C4.0, дающая преимущество по скорости работы и используемой памяти, строящая деревья, сравнимые по эффективности, но меньшего размера.
- CART [22] (Classification and Regression Trees) строит бинарное дерево решений с использованием критерия Джини.

Деревья принятия решений и решения, построенные на их основе, являются одним из самых популярных способов решения задачи классификации трафика, так как среди их достоинств можно перечислить:

- отсутствие необходимости в специальной подготовке данных (нормализация, добавление фиктивных переменных для приведения примеров к единому размеру и др.) как при работе с нейронными сетями (см. [23]),
- способность работы с разными типами переменных (как категориальными, так и интервальными),
- способность работать с большими объёмами данных,
- хорошие результаты классификации [11, 18],
- высокую скорость работы и низкую вычислительную сложность предсказания результата построенным деревом [18],
- простоту мультиклассовой классификации (в сравнении, например, с SVM).

К недостаткам этого метода можно отнести:

- неоптимальность алгоритмов построения дерева (проблема получения оптимального дерева решений является NP-полной, поэтому оптимальное решение выбирается локально в каждом узле, что влияет на оптимальность дерева в целом),
- риск переобучения (необходимо регулировать глубину дерева),
- сложность выражений некоторых ситуаций посредством дерева (например, XOR),
- неустойчивость деревьев при небольших изменениях входных данных.

С этими недостатками можно бороться применением алгоритмов бэггинга и бустинга, которые рассматриваются далее.

4.5 Бэггинг

Бэггинг (bagging от bootstrap aggregating) – это способ композиции нескольких более простых классификаторов в один с целью повысить его стабильность и точность. В частности, широко используется алгоритм *случайный лес* (Random Forest) [9, 11, 13, 24, 25], заключающийся в использовании комитета (ансамбля) решающих деревьев. Основная идея этого метода в

использовании большого ансамбля деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим. Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев.

В приведённых статьях метод бэггинга выбирается как основной для классификации или показывает одни из лучших результатов в сравнении с другими методами. Получающиеся модели сложнее, чем для рассмотренных выше классификаторов, а время на классификацию одного примера растёт примерно линейно относительно количества простых классификаторов в ансамбле, однако именно этот метод зачастую даёт наилучший вариант компромисса между качеством и скоростью классификации.

4.6 Бустинг

Бустинг (boosting) – способ организации классификаторов, позволяющий упорядочить и обучить более простые классификаторы (чаще всего это решающие деревья небольшой глубины) таким образом, чтобы результирующий классификатор показывал лучшие результаты.

Примерами алгоритмов бустинга являются:

- AdaBoost (Adaptive Boosting) [13, 18, 26] – регулирует веса в процессе обучения, чтобы примерам, на которых ошибся прошлый классификатор, придавалось большее значение;
- XGBoost [24] – ошибка минимизируется алгоритмом градиентного спуска.

По качеству и скорости работы бустинг примерно сравним с бэггингом (зависит от конкретных данных; один из примеров такого сравнения можно найти в [13]).

Можно видеть, что из рассмотренных на данный момент моделей машинного обучения для классификации сетевого трафика чаще всего и с наибольшим успехом используются деревья принятия решения (с разными алгоритмами построения), а также их комбинации методами бэггинга или бустинга.

4.7 Нейронные сети

Искусственные нейронные сети состоят из нескольких слоёв искусственных нейронов, каждый из которых получает на вход несколько числовых значений и преобразует их в выходное значение в соответствии со своими внутренними правилами (заранее заданной функцией активации и вычисляемыми весами входных параметров). Слой сети может состоять из произвольного числа нейронов, каждый из которых может быть соединён с любыми нейронами из предыдущего и последующего слоёв.

Существуют модели так называемых рекуррентных нейронных сетей с обратной связью, когда сигнал с выходных нейронов или нейронов скрытого слоя частично передается обратно на входы нейронов входного слоя, но чаще всего нейронные сети являются сетями прямого распространения (feedforward). Значения нейронов в каждом слое вычисляются на основании значений предыдущего слоя в ходе процесса, называющегося алгоритмом прямого распространения (forward propagation). В процессе обучения изменение параметров направлено на минимизацию функции штрафа (cost function). Для коррекции весов в процессе обучения применяется алгоритм обратного распространения ошибки (backpropagation).

Для работы с большим количеством параметров во внутренних слоях сети или для определения инвариантных относительно переноса признаков свою эффективность показали свёрточные нейронные сети (CNN), использующие набор небольших ядер для преобразования поступающей информации и уменьшения числа извлекаемых признаков [12, 23, 27-30].

Рекуррентные сети (RNN), как уже упоминалось ранее, могут использоваться для работы с признаками, имеющими зависимости во времени (например, в случае, когда текущий ответ сети должен зависеть не только от текущих признаков, но и от некоторых признаков, использовавшихся при принятии предыдущего решения). Для этого сеть особым образом сохраняет часть извлекаемой информации для дальнейшего использования. Самым часто используемым представителем является LSTM (Долгая краткосрочная память от англ. Long short-term memory) [27, 28].

Автокодировщики (Autoencoders) представляют из себя нейронные сети, состоящие из двух частей: первая половина сети учится кодировать входную информацию в сжатом виде, а вторая учится с максимально возможной точностью воссоздавать эту информацию по сжатому представлению. Поскольку получаемый метод кодирования эффективно работает только для того типа данных, на котором обучалась сеть, другие данные будут воссоздаваться с ошибкой, что позволяет использовать автокодировщики для классификации [12, 31, 32].

Генеративно-состязательная сеть (GAN, Generative adversarial network) - комбинация из двух нейронных сетей, одна из которых учится генерировать образцы данных, похожие на реальные, а вторая - отличать эти образцы. В процессе совместного обучения качество работы обеих этих сетей растёт, что позволяет генерировать искусственные данные, почти неотличимые от реальных. Такой подход может использоваться для генерации дополнительных примеров при невозможности получить их достаточное количество другим путём [9].

Также существуют примеры совместного использования нескольких типов нейронных сетей для получения лучших результатов, в частности, CNN+RNN [14, 27, 33].

5. Признаки сетевого трафика, используемые для его классификации

Алгоритмы машинного обучения для своей работы нуждаются в получении признаков классифицируемых примеров, на основе которых будет приниматься решение. Для задачи классификации сетевого трафика можно выделить два основных способа их получения:

- признаки выделяются на специальном этапе подготовки данных на основе некоторой внешней информации (например, знания эксперта в области); такие признаки могут включать в себя как содержимое пакетов, так и дополнительную информацию о потоке (такую, как общее количество пакетов, размер переданных данных, время получения пакетов и т.п.);
- признаки выделяются из данных (обычно это содержимое пакетов) самой моделью в процессе так называемого глубокого обучения.

Первый подход требует от экспериментатора дополнительных усилий, но позволяет лучше контролировать процесс и использовать любую доступную информацию и её производные (например, посчитанные на основе имеющихся данных статистики). Второй подход требует минимальных действий при подготовке данных (нормализовать данные, унифицировать длину примеров, переупорядочить данные, если это необходимо) и может позволить находить неочевидные на первый взгляд зависимости, но извлекаемый им набор признаков не всегда является лучшим или минимальным для решения поставленной задачи. Кроме того, второй подход обычно требует для обучения большего количества данных.

По содержанию признаков их можно несколько условно разбить на следующие классы:

- данные пакета;
- метаинформация о пакете;
- временные характеристики;
- информация о потоке.

Первая категория включает в себя содержимое полезной нагрузки пакета или все байты пакета без из разделения на заголовок и данные. Именно она чаще всего используется при

глубоком обучении модели классификации [12, 23, 27]. Выделяемые из содержимого пакетов закономерности очень информативны (такой подход является в некотором смысле автоматизированным аналогом DPI). В [12] показано, что такой метод может применяться и для классификации зашифрованного трафика, однако количество информации о пакете, используемой для его классификации достаточно велико (1480 байтов). В [27] поднимается вопрос о возможности переобучения сети под искусственно сгенерированный набор данных: при таком наборе признаков модель способна запоминать IP-адреса или номера портов, которые используются для разных классов. В таком случае показанные ей в экспериментах результаты будут высокими, но такая модель не будет применима для реальных сетей. Эта возможность не учтена в [23], где используются 784 первых байтов пакета, включая все уровни заголовка.

Под метаинформацией о пакете можно понимать такие признаки, как размер пакета, размер его полезной нагрузки, направление движения, а также тип сервиса, указанный тип протокола, установленные флаги и размер окна (для TCP). Различные комбинации этих признаков используются во многих работах по теме.

Временные характеристики включают в себя интервалы времен между прибытием пакетов (этот признак весьма показателен для классификации трафика по разным сценариям использования), общее время сессии. Также, в статье [21] предложена идея использования временных вспышек трафика. Временная вспышка (burst) трафика – это группа последовательных пакетов с интервалами прибытия между ними меньше, чем между разными вспышками. Характеристики вспышек в потоке трафика (burstiness) являются мерой изменчивости распределения времён прибытия пакетов. В новом варианте исследований [10] от тех же авторов был добавлен ещё такой признак, как время бездействия потока.

Информация о потоке объединяет в себе метаинформацию всех или части пакетов (в целом или в каждом направлении отдельно) в потоке в виде некоторых её статистических характеристик: сумма по всех пакетам, минимум, максимум, среднее арифметическое, медиана, дисперсия и т.д. Также, могут вычисляться аналогичные характеристики для временного распределения пакетов. Эти признаки дополняют и расширяют информацию, получаемую из двух предыдущих категорий признаков, но требуют для своего вычисления завершения потока, что затрудняет их использование в режиме классификации в реальном времени. Для решений этой проблемы можно использовать не все, а только N первых пакетов в потоке (например, [23]. Значение N подлежит экспериментальному вычислению или выбирается на основе экспертной оценки.

Итак, разные исследования задачи классификации трафика используют разные категории признаков, одну или несколько из перечисленных выше. Задача выбора и отбора признаков весьма актуальна и решается практически в каждом новом исследовании. Существуют статьи [34] и инструменты [35], посвящённые описанию и получения максимально возможного количества доступных признаков для пакетов в потоке данных.

6. Выбор и подготовка набора данных, их сравнение

Препятствием для прямого сравнения различных подходов в работах исследовательских групп по классификации трафика является не только наличие нескольких разных принципов классификации (по протоколам/приложениям/типам приложений/...), но и отсутствие единого общепризнанного набора данных, на которых бы проводилось тестирование предлагаемых методов. На настоящий момент отсутствие такого стандартного набора означает, что каждая исследовательская группа самостоятельно находит/собирает данные для обучения и тестирования моделей, размечает их по своей системе своими силами и способами и публикует полученные на них результаты. Это не позволяет сопоставлять эти результаты между собой напрямую, как делается при решении многих типовых задач, что является существенным недостатком в данной области.

Получение большой и репрезентативной выборки данных для обучения и тестирования моделей – первая проблема, которая возникает при решении задачи классификации данных. Каждая исследовательская группа должна найти для себя подходящий источник данных, убедиться в корректности его разметки соответственно поставленной задаче и оценить его полноту относительно возможных вариантов организации трафика в сети. Количество существующих приложений/протоколов огромно, поэтому производить полную классификацию вряд ли представляется возможным. Обычно исследователи выбирают лишь некоторый набор из возможных вариантов (наиболее частые/характерные/представляющие наибольший интерес) и работают только с ним. Однако в таком случае возникает проблема обработки данных, не входящих в данный набор классов. В некоторых работах эти данные просто не рассматриваются, но этот подход неизбежно сталкивается с проблемами при работе с данными реального мира. Ещё одной проблемой является частое появление новых приложений/протоколов, которые либо должны своевременно отражаться в данных, либо обрабатываться особым образом. Также, стоит учитывать, что от метода сбора данных зависит не только их репрезентативность, но и распределение вероятностей классов, что также может влиять на конечный результат.

6.1 Основные аспекты выбора набора данных

При выборе или получении данных особое внимание следует уделять следующим аспектам.

6.1.1 Получение правильной разметки данных

При применении систем, использующих методы машинного обучения для классификации сетевого трафика, помимо тестовой выборки для оценки результатов нужна достаточно большая тренировочная выборка с правильно размеченными ответами. Соответственно, возникает проблема получения этой разметки. Источников здесь может быть несколько.

a) *Сторонние системы классификации.* Например, можно использовать методы, разбирающие и анализирующие всю информацию, содержащуюся в пакете, для построения более легковесных и оперативных классификаторов. Под это определение подходят системы DPI, такие как Wireshark [36], nDPI [37] и др. Сравнение некоторых доступных для использования систем DPI произведено в [38, 39]. Для проверки правильности разметки инструментов предлагается протестировать их работу на множестве данных, для которых специальная программа регистрирует генерирующие их приложения, что позволяет достичь заведомо высокого качества разметки. Исследование показывает, что использование инструментов DPI даёт высокую, но не идеальную точность при определении приложений.

   Соответственно, при этом подходе, следует учитывать, что точность такой разметки, а соответственно, и обучаемых на ней алгоритмов, ограничена точностью этих сторонних методов. Кроме того, этот способ может быть неприменим при работе с зашифрованным трафиком.

b) *Получение данных в контролируемых условиях.* В этом случае нужно организовать сбор информации как можно ближе к пользователю и поставить перед ним задачу генерировать только заранее определённый трафик. Самая распространённая проблема в этом случае – фильтрация фонового трафика, доля которого может доходить до 70%.

   Для контроля за получением трафика можно либо собирать его в процессе исполнения специальных скриптов, генерирующих только определённые его классы (ISCX, NIMS), либо запускать параллельно со сбором трафика специальные программы, позволяющие определить его источник (UPC dataset).

c) *Генерация искусственных данных на основе существующих.* Ещё один способ получения дополнительных данных для тех классов, в которых этих данных недостаточно для обучения модели, – это дополнение тренировочного множества искусственными

данными. Например, в [9, 28] для этой цели используется LSTM. Как показано, такой метод позволяет улучшать качество построенной модели, особенно в тех случаях, когда некоторые из классов недостаточно представлены в тренировочном множестве. Однако, для его использования нужен хорошо обученный генератор данных, что является отдельной задачей, которую также надо решать. Кроме того, нужно следить, чтобы получаемые таким образом данные не были излишне однообразными и соответствовали реальному положению дел в сети.

6.1.2 Доступность данных

Из-за шифрования данных или в связи с необходимостью соблюдения конфиденциальности пользователей, часть данных в пакетах может быть искажена или недоступна. Например, при публикации снимков сетевых трасс может производиться специальное кодирование IP адресов или удаление полезной нагрузки пакетов. Подобные действия могут создавать трудности, особенно при разметке данных, однако этический вопрос в данном случае должен иметь преимущество. Одним из способов поиска компромисса в данной ситуации является сбор статистики поведения реальных пользователей сети интернет и написание специальных алгоритмов-ботов, которые будут имитировать это поведение с максимальной правдоподобностью, при этом не выдавая никакую конфиденциальную информацию.

Такой подход к решению проблемы получил достаточно широкое распространение и позволил получить содержательные датасеты для многих задач, связанных с анализом трафика в сети. Одним из его существенных недостатков является, однако, большая упорядоченность и детерминизм действий по сравнению с реальными пользователями, что может нарушать естественное распределение статистических признаков потоков данных.

6.1.3 Место получения данных

Получаемые для анализа потоки данных зависят от места их сбора. Получение данных как можно ближе к конечным пользователям позволяет проще решать вопрос с шифрованием данных, в то время как сбор информации у провайдера, на глобальных маршрутизаторах позволяет получить больший объём разнообразных и репрезентативных данных. В то же время, такие особенности, как изменение временного распределения в потоке, изменение длины пакетов при туннелировании, наличие на маршрутизаторе, служащем точкой сбора информации, только одного направления потока данных, - все они могут приводить к тому, что модель, обученная на данных из одного источника, может не работать на данных из другого места. Поэтому, предпочтительной стратегией в данном случае является использование в качестве тренировочных данных, получаемых из тех же точек в сети, где предполагается работа создаваемой системы.

В [40] исследуется влияние особенностей характеристик сетей на работу классификатора, в частности показано падение результатов работы классификатора при его тестировании на трафике из сети, отличной от той, на которой он обучался.

6.1.4 Репрезентативность набора данных

После выбора системы используемых классов и разметки данных требуется убедиться в том, что полученный набор данных является репрезентативным, то есть содержит достаточное количество разнообразных примеров для каждого класса (по возможности, покрывает все возможные ситуации). Следует нивелировать перекос по количеству данных для разных классов для предотвращения переобучения классификатора. Некоторые источники получения данных требуют особого внимания и предобработки на данном этапе. Например, если набор данных получен только от нескольких пользователей, то нужно постараться убедиться, что модель сможет выделить из данных глобальные признаки, а не будет пытаться определить специфику работы каждого конкретного пользователя.

6.2 Используемые общедоступные наборы данных

Поскольку сам процесс получения большого количества релевантных данных, пусть и неразмеченных, может быть затруднителен, в исследованиях нередко используются широко доступные трассы сетевого трафика, которые можно найти в интернете. Обычно такие трассы содержат достаточно большой объём данных и различаются по месту и способу их получения, а также предоставленной разметке (если она есть) К наиболее популярным из общедоступных наборов данных сетевого трафика можно отнести следующие.

- *MooreSet* (2007) [18]. Около 59 Гигабайтов данных. Данные были собраны на границе сети университетского кампуса в течение 8 месяцев. Набор данных содержит только потоки TCP. Большая их часть классифицирована вручную на основе содержания на 10 классов, также присутствуют некоторые фоновые потоки и потоки без начала. Классы: web-browsing, mail, bulk, attack, p2p, database, multimedia, service, interactive, games.
- *UPC* (Политехнический университет Каталонии) dataset (2013) [38]. Около 36 Гб данных, собранных за 66 дней. Ради возможности публикации данных с полной полезной нагрузкой трафик был сгенерирован искусственно, авторы постарались максимально имитировать реальный трафик. Для 91% пакетов и 42% потоков присутствует название приложения, полученное с помощью специальной программы, записывающей информацию о каждом сетевом потоке, включая название приложения.
- *CAIDA* (2008-...). Содержит набор анонимизированных трасс, собранных на мониторах магистральных сетей связи. С 2008 по 2014 год в набор добавлялась одна часовая трасса в месяц, с 2014 года – раз в три месяца. Из пакетов удалена полезная нагрузка, IP-адреса зашифрованы инструментом Crypto-PAn (Cryptography-based Prefix-preserving Anonymization) [41]. Размер каждой трассы – несколько миллиардов пакетов.
- *ISCX* (2016). Содержит несколько трасс с искусственно сгенерированными данными для решения разных задач сетевого трафика (поиск аномального поведения, поиск ботнетов, классификация трафика и т.д.). Данные сгенерированы так, чтобы по возможности имитировать поведение реальных пользователей. Наиболее популярным набором данных в исследованиях по классификации трафика является VPN-nonVPN (ISCXVPN2016) [42]. Эта трасса содержит 7 категорий трафика: браузинг, электронная почта, чаты (мгновенные сообщения), стриминг, передача файлов, VoIP, p2p. При этом каждая категория представлена в двух видах - через VPN и без него. Полезная нагрузка пакетов сохранена, общий объём файлов составляет 28Гб.
- *NIMS* (2007). Этот набор данных был собран в специально собранной модели сети посредством прописанных сценариев её использования. Основной целью сбора трафика являлось получение SSH трафика, а в качестве фонового были собраны DNS, HTTP, FTP, P2P (limewire) и telnet. Всего датасет содержит около 700 000 потоков, из них около 35000 – SSH потоков.

Таким образом, единого набора данных для решения задач классификации трафика не существует, и каждая исследовательская группа должна найти или подготовить данные в соответствии со своими целями. При принятии решения о выборе данных для обучения и тестирования классификатора нужно учитывать приведённые выше соображения и ограничения, чтобы полученная модель соответствовала построенной задаче и была способна работать в реальных условиях.

7. Заключение: актуальные проблемы

7.1 Создание общедоступных датасетов и защита данных пользователей

Поскольку используемые и анализируемые данные, составляющие полезную нагрузку пакетов, являются приватной информацией пользователей интернета, работы в этой области должны придерживаться определённых моральных норм. Защита личной жизни пользователей и конфиденциальность их информации должна являться главным приоритетом для исследователей. Одним из аспектов этого вопроса является процесс получения данных, в частности данных для обучения моделей машинного обучения, которые заведомо должны содержать больший объём доступной для исследования информации, так как для них требуется правильно определить их принадлежность к определённым классам. Хранение и обработка этих данных должны обеспечивать безопасность личных данных пользователей и их анонимность.

Однако при этом, как и в других областях, в которых активно развиваются методы решения на основе машинного обучения, желательны создание и поддержка в актуальном состоянии достаточно больших и универсальных наборов правильно размеченных данных, которые могли бы использоваться для проверки работы и сравнения качества разных предлагаемых методов решения задач. Общепринятого компромисса о том, как организовать такой набор с соблюдением защиты информации пользователей, до сих пор нет, хотя и делались попытки предложить варианты решения этой проблемы.

Одним из предложенных решений является анонимизация трафика; как частный её пример – эквивалентная замена адресов во всех используемых пакетах [41]. Так как сами IP-адреса редко имеют какую-либо ценность при классификации трафика, предлагается шифровать их таким образом, чтобы каждому значению однозначно сопоставлялся IP-адрес, но при этом это адрес невозможно было бы восстановить по получаемому значению. Такой подход позволяет защитить конфиденциальность пользователей, не позволяя сопоставить наборы потоков данных с конкретными людьми по их IP-адресам. Но он не решает проблему в случае, если по сети передаются чувствительные данные, такие как пароли, номера кредитных карт и т.п.

В некоторых общедоступных наборах данных с этим борются удалением всей полезной нагрузки из пакета (оставляя только заголовки пакета). Этот способ плох тем, что затрудняет или даже делает невозможной предварительную разметку данных (недостаточно информации для систем DPI). Некоторые данные выкладываются параллельно с такой разметкой, полученной авторами датасета до модификации данных или даже в процессе их получения [38]. Но приведённая разметка может использовать другой набор классов, чем хотелось бы получить с использованием приведённых данных, поэтому такой способ также нельзя считать универсальным.

Широкое распространение получило создание и предоставление в общий доступ наборов искусственно сгенерированных данных [38]. Авторы таких датасетов анализируют трафик реальных пользователей и на основе результатов этого анализа создают ботов, которые генерируют трафик с заданными определёнными свойствами. Получаемые данные имитируют существующий трафик в сети, но при этом не компрометируют реальных людей и не содержат личных и чувствительных данных. Здесь, опять же, возникает вопрос соответствия такого искусственного трафика и реального: все ли особенности были учтены, не являются ли искусственные данные излишне детерминированными, насколько все их характеристики, которые могут использоваться алгоритмами машинного обучения для выделения закономерностей, отражают реальную ситуацию, а насколько определяются скриптами генерации.

Единого ответа на все эти вопросы, которые нужно решить для создания общего универсального набора данных для задачи классификации трафика, на данный момент не существует. Однако для более эффективной работы в данной области нужно искать пути решения этой проблемы.

7.2 Классификация с использованием частей потоков, классификация по середине потока

Согласно исследованиям, наиболее хорошие результаты показывают системы, использующие для классификации потоков информацию о его первых пакетах. Также существует достаточное количество исследований, показывающих, что для классификации потоков не обязательно дожидаться их окончания, а достаточно использовать первые N пакетов. Однако мы не всегда можем поймать или зафиксировать начало потока. Вдобавок, для многих практических приложений требуется только классификация потоков достаточной длины, а многие из слишком коротких потоков являются ошибочными. В таком случае желательно было бы не начинать сохранять информацию о потоке до того момента, как мы увидим хотя бы несколько пакетов из него, для экономии памяти системы. На данный момент нет достаточного количества полных исследований, показывающих, насколько эффективной может быть классификация потоков в случае использования произвольной части пакетов из их середины и применимо ли данное предложение в реальной жизни.

7.3 Работа с новыми классами данных

Так как новые приложения и протоколы продолжают появляться регулярно, становится невозможно в любой момент времени поддерживать базы данных и классификаторы в актуальном состоянии (нужно зафиксировать момент появления нового класса, собрать достаточное количество данных для него и переобучить классификаторы - всё это займёт время, даже если есть возможность производить эти действия в автоматическом режиме). Распределение примеров только по набору известных классов также не является решением, поэтому возникает проблема выделения в классификаторе неизвестного класса (классов). В идеале система должна уметь объединять новые неизвестные для себя объекты в один класс (например, с помощью алгоритмов кластеризации) и запрашивать для него метку, однако даже более простую задачу выделения всех неизвестных примеров в один общий класс нельзя считать окончательно решённой.

7.4 Использование полученных моделей в других сетях

Как уже было отмечалось выше, выбор данных для обучения очень важен при построении модели, и результаты работы классификатора в сети или в месте сети, отличном от того, где он обучался, ухудшает качество его работы [40]. Для борьбы с этой проблемой нужно больше исследований относительно того, сколько данных необходимо для переобучения модели под другую сеть, как меняется точность классификации со временем, можно ли автоматически отслеживать необходимость в дообучении, и насколько можно автоматизировать этот процесс.

Список литературы / References

[1]. Rezaei S., Liu X. Deep learning for encrypted traffic classification: An overview. IEEE Communications Magazine, vol. 57, issue 5, 2019, pp. 76-81.

[2]. Jamshidi S. The Applications of Machine Learning Techniques in Networking. Available at: https://www.cs.uoregon.edu/Reports/AREA-201902-Jamshidi.pdf, accessed 30.10.2020.

[3]. Hubballi N., Swarnkar M. BitCoding: Network Traffic Classification Through Encoded Bit Level Signatures. IEEE/ACM Transactions on Networking, vol. 26, issue 5, 2018, pp. 1-13.

[4]. Hubballi N., Swarnkar M., Conti M. BitProb: Probabilistic Bit Signatures for Accurate Application Identification. IEEE Transactions on Network and Service Management, vol. 17, no. 3, 2020, pp. 1730-1741.
[5]. Finamore A., Mellia M., Meo M., Rossi D. KISS: Stochastic Packet Inspection Classifier for UDP Traffic. IEEE/ACM Transactions on Networking, vol. 18, no. 5, 2010, pp. 1505-1515.
[6]. Dorfinger P., Panholzer G., John W. Entropy estimation for real-time encrypted traffic identification. In Proc. of the Third international conference on Traffic monitoring and analysis (TMA'11), 2011, pp. 164-171.
[7]. Khakpour A.R., Liu A.X. High-Speed Flow Nature Identification. In Proc. of the 29th IEEE International Conference on Distributed Computing Systems, 2009, pp. 510-517.
[8]. Doroud H., Aceto G. et al. Speeding-Up DPI Traffic Classification with Chaining. In Proc. of the IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1-6.
[9]. Vu L., Bui C.T., Nguyen Q.U. A Deep Learning Based Method for Handling Imbalanced Problem in Network Traffic Classification. In Proc. of the Eighth International Symposium on Information and Communication Technology, 2017, pp. 333-339.
[10]. Oudah H., Ghita B., Bakhshi T. A Novel Features Set for Internet Traffic Classification using Burstiness. In Proc. of the 5th International Conference on Information Systems Security and Privacy, vol. 1, 2019, pp. 397-404.
[11]. Aceto G., Ciuonzo D., Montieri A., Pescapé A. Multi-classification approaches for classifying mobile app traffic. Journal of Network and Computer Applications, vol. 103, 2018, pp. 131-145.
[12]. Lotfollahi M., Jafari Siavoshani M., Shirali Hossein Zade R. et al. Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Computing, vol. 24, issue 3, 2020, pp. 1999-2012.
[13]. Gómez S.E., Martínez B.C. et al. Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal. Computer Networks, vol. 127, 2017, pp. 68-80.
[14]. Lopez-Martin M., Carro B., Sanchez-Esguevillas A., Lloret J. Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things. IEEE Access, vol. 5, 2017, pp. 18042-18050.
[15]. Mercaldo N., Lu W. Classification of Web Applications Using AiFlow Features. In Proc. of the Workshops of the International Conference on Advanced Information Networking and Applications, 2020, pp. 389-399.
[16]. Wang P., Chen X., Ye F., and Sun Z. A survey of techniques for mobile service encrypted traffic classification using deep learning. IEEE Access, vol. 7, 2019, pp. 54024-54033.
[17]. Takyi K., Bagga A., Gupta P. A Semi-Supervised QoS-Aware Classification for Wide Area Networks with Limited Resources. International Journal of Innovative Technology and Exploring Engineering, vol. 8, issue 11, 2019, pp. 970-981.
[18]. Li W., Moore A. W. A Machine Learning Approach for Efficient Traffic Classification. In Proc. of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007, pp. 310-317.
[19]. Ding Y. A method of imbalanced traffic classification based on ensemble learning. In Proc. of the IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), 2015, pp. 1-4.
[20]. Carela-Español V. et al. Analysis of the impact of sampling on NetFlow traffic classification. Computer Networks, vol. 55, issue 5, 2011, pp. 1083-1099.
[21]. Oudah H., Ghita B., Bakhshi T. Network application detection using traffic burstiness. In Proc. of the World Congress on Internet Security, 2017, pp. 148-152.
[22]. Soylu T., Erdem O., Carus A. Bit vector-coded simple CART structure for low latency traffic classification on FPGAs. Computer Networks, 2020, vol. 167, article id 106977.
[23]. Wang W., Zhu M., Wang J., Zeng X., Yang Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proc. of the IEEE International Conference on Intelligence and Security Informatics (ISI), 2017, pp. 43-48.
[24]. Zhao S., Chen S. et al. Identifying Known and Unknown Mobile Application Traffic Using a Multilevel Classifier. Security and Communication Networks, vol. 2019, 2019, article id 9595081.
[25]. Brissaud P., Francçis J., Chrisment I., Cholez T., Bettan O. Transparent and Service-Agnostic Monitoring of Encrypted Web Traffic. IEEE Transactions on Network and Service Management, vol. 16, no. 3, 2019, pp. 842-856.

[26]. Jin Y., Duffield N. et al. A modular machine learning system for flow-level traffic classification in large networks. ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 6, issue 1, 2012, pp. 1-34.
[27]. Rezaei S., Kroencke B., Liu X. Large-Scale Mobile App Identification Using Deep Learning. IEEE Access, vol. 8, 2020, pp. 348-362.
[28]. Hasibi R., Shokri M., Dehghan M. Augmentation scheme for dealing with imbalanced network traffic classification using deep learning. arXiv preprint, arXiv:1901.00204, 2019.
[29]. Zhao L. et al. A novel network traffic classification approach via discriminative feature learning. In Proc. of the 35th Annual ACM Symposium on Applied Computing, 2020, pp. 1026-1033.
[30]. Rezaei S., Liu X. How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets. arXiv preprint, arXiv:1812.09761, 2018.
[31]. Wang Z. The applications of deep learning on traffic identification. BlackHat USA, vol. 24, issue 11, 2015, pp. 1-10.
[32]. Zheng W., Gou C., Yan L., Mo S. Learning to Classify: A Flow-Based Relation Network for Encrypted Traffic Classification. In Proc. of the Web Conference, 2020, pp. 13-22.
[33]. Zeng Y., Qi Z. et al. TEST: an End-to-End Network Traffic Examination and Identification Framework Based on Spatio-Temporal Features Extraction. arXiv preprint, arXiv:1908.10271, 2019.
[34]. De Montigny-Leboeuf A. Flow attributes for use in traffic characterization. Technical Note CRC-TN-2005-00, Communications Research Centre Canada, 2005.
[35]. Burschka S., Dupasquier B. Tranalyzer: Versatile high performance network traffic analyser. IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1-8.
[36]. Orebaugh A., Ramirez G., Beale J. Wireshark & Ethereal network protocol analyzer toolkit. Elsevier, 2006, 448 p.
[37]. Deri L. et al. ndpi: Open-source high-speed deep packet inspection. In Proc. of the IEEE International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 617-622.
[38]. Carela-Español V., Bujlow T., Barlet-Ros P. Is our ground-truth for traffic classification reliable? Lecture Notes in Computer Science, vol. 8362, 2014, pp. 98-108.
[39]. Bujlow T., Carela-Español V., Barlet-Ros P. Independent comparison of popular DPI tools for traffic classification. Computer Networks, vol. 76, 2015, pp. 75-89.
[40]. Khatouni A. S., Heywood N. Z. How much training data is enough to move a ML-based classifier to a different network? Procedia Computer Science, vol. 155, 2019, pp. 378-385.
[41]. Fan J., Xu J., Ammar M. H., Sue M. Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-Based Scheme. In Proc. of the 10th IEEE International Conference on Network Protocols (ICNP 2002), 2002, pp. 12-15.
[42]. Draper-Gil G., Lashkari A.H., Mamun M.S.I., Ghorbani A.A. Characterization of encrypted and VPN traffic using time-related. In Proc. of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407-414.

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 – PhD 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.

Мария Кирилловна ИКОННИКОВА – аспирант. Научные интересы: анализ сетевого трафика, машинное обучение.

Maria Kirillovna IKONNIKOVA – postgraduate student. Research interests: network traffic analysis, machine learning.

DOI: 10.15514/ISPRAS–2020–32(6)–12

Automated Analysis of DP-systems Using Timed-Arc Petri Nets via TAPAAL Tool

[1] *A.A. Izmaylov, ORCID: 0000−0003−1721−3347 <zinoviy23@gmail.com>*
[1] *L.W. Dworzanski, ORCID: 0000−0002−0074−7660 <leo@mathtech.ru>*

[1] *National Research University Higher School of Economics, Myasnitskaya st., 20, Moscow, 101000, Russia*

Abstract. Timed-Arcs Petri nets (TaPN-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 and motivated by study of localized Gaussian wave packets scattering on thin structures; as well, DP-systems could be utilized to overapproximate the dynamics of messages scattering in distributed systems. In the latter case, time-temporal properties of DP-systems become a matter of interest. However, there are no tools that enable us to analyse them. In this work, we provide a new approach to automated analysis of DP-systems using the translation of a DP-system into a TaPN-net which is implemented as a TAPAAL plugin. The translation let us use the comprehensive tool support for TaPN-nets (TAPAAL/UPPAAL) to analyze DP-systems dynamical characteristics expressed in TCTL language. We demonstrated how to express some of them and verify time-temporal properties of a DP-system using the suggested approach, and performed experiments to obtain empirical estimates of the tool performance.

Keywords: metric graphs; timed-arc Petri nets; time temporal properties

For citation: 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

Acknowledgements. This work was supported by the RFBR grant 20-07-01103 a.

Автоматический анализ дискретных динамических систем на метрических графах с помощью сетей Петри с временными дугами и инструмента TAPAAL

[1] *А.А. Измайлов, ORCID: 0000−0003−1721−3347 <zinoviy23@gmail.com>*
[1] *Л.В. Дворянский, ORCID: 0000−0002−0074−7660 <leo@mathtech.ru>*

[1] *Национальный исследовательский университет «Высшая школа экономики», Россия, 101000, г. Москва, ул. Мясницкая, д. 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; кроме того, DP-системы могут использоваться для приближенного представления динамики распространения сообщений в распределенных системах. В этой работе, мы описываем новый подход для автоматического анализа DP-систем используя трансляцию в TaPN сеть, которая реализована как расширение инструмента TAPAAL. Подход позволяет использовать мощные инструменты верификации (TAPAAL/UPPAAL) для анализа динамических характеристик DP-систем,

представленных на языке TCTL. В работе продемонстрировано, как можно кодировать временно-темпоральные свойства DP-систем в рамках предложенного подхода, и приведены результаты экспериментальных тестов.

Ключевые слова: метрические графы; сети Петри с временными дугами; темпорально-временные динамические свойства

Для цитирования: Измайлов А.А., Дворянский Л.В. Анализ дискретных динамических систем на метрических графах с помощью сетей Петри с временными дугами и инструмента TAPAAL. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 155-166 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(6)–12

Благодарности: Работа выполнена при поддержке гранта РФФИ 20-07-01103 а.

1. Introduction

The notion of a Petri nets evolved from a chemistry process model to a model of indefinitely expandable computing system consistent with laws of physics in C.A. Petri works [1]. Nowadays, 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]. Timed-arc Petri nets (TaPN-nets) are an extension of Petri nets with time semantics: tokens are assigned clocks [2]; an inscription on an incoming arc of a transition define tokens of which age can be consumed by the firing of the transition.

Metric graphs are graphs with lengths assigned to edges. A dynamical system consisting of a metric graph and dynamic points moving along the graph edges (DP-system) is a geometrical discrete dynamical system originally motivated by the problem of evolution of wave packets in thin structures. It could be considered as a simplified discrete model of a quantum graph – a metric graph equipped with functions on its edges, a differential operator acting on such functions, and matching conditions on its vertices [3], [4] – with narrow localized wave packets [5], [6]. Quantum graphs occurred as a model or tool in a number of problems in chemistry, physics, engineering, and mathematics since 1930s [7]. It was shown that there exists a correspondence between the statistics of localized solutions on a quantum graph and the dynamics of a DP-system [6]. Points in a DP-system may represent supports of Gaussian wave packets in a quantum graph and/or the projection of wave propagation on medium geodesics. Both models, Petri nets and DP-systems, embrace real-time dynamics of discrete entities moving within a topological structure defined by a graph.

Some results towards the characteristics of the dynamics of DP-systems were recently obtained in [8–11]. The growth of the number of points moving along edges and its asymptotic are studied for metric trees in [10] and, for some special cases, in [6]. In [11], polynomial approximation for the growth of the number of dynamic points moving along metric graphs is studied, and explicit formulae for the first two terms of the polynomial approximation are given. In [6], stabilization of the number of points in a DP-system is studied and explicit formulae for graphs with edges of the same length and star graphs are given, exploiting a connection with analytic number theory problems. While the mentioned quantitative dynamic statistics of DP-systems are studied, finer temporal properties of DP-systems are not; like, for example, which node v_1 or v_2 will receive a point first, or whether e_1 will have received ten or more points by the time e_2 received its first point. In this work, we provide a translation of a DP-system into a TaPN-net [12], which allows reducing the analysis of temporal properties of DP-system to analysis of TaPN-net and conduct it using the effective widely-used TAPAAL tool [13], which is under active development (two golds, a silver, and a bronze medal in MCC'20 [14]).

Temporal properties of Petri nets, such as liveness, boundedness, and reversibility, and their complexities have been extensively studied (see comprehensive review [15]); and, many of them are in PSPACE or, worse, in EXPSPACE complexity classes. The study and development of Petri nets analysis methods is still active; for example, the long-standing question on whether reachability and coverability problems have the same complexity has been recently resolved in [16] showing that

reachability is not even elementary, thus, impacting complexity of many problems in other fields, such as formal languages, logic, linear algebra, etc. However, for some restricted Petri net subclasses, many important properties, which are undecidable in general case, are decidable or even have polynomial time complexity.

There is a lot of ways to introduce time constraints in Petri nets [17-18]. Unfortunately, it was shown that almost any of semantical extensions makes Petri nets Turing-complete and, by Rice-Uspensky theorem, many of general behavioural problems immediately become undecidable. The widely known time extensions of Petri nets – Time Petri nets [19] and Timed (Duration) Petri nets [20] – 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 [21-23]. A time semantics with restricted urgency was recently suggested for TaPN-nets in [24]; the suggested semantics allows urgent transitions to consume tokens only from the bounded places of a Petri net, and this restriction makes some behavioural problems decidable for TaPN-nets. In [25], author suggested an approach to use timing specifications to improve the behavioural properties of an untimed P/T-net with an example of making live and unbounded untimed P/T-net live and bounded by cutting off token generators using time-based constraints.

TaPN-nets attract our attention as they feature dynamics that makes it possible to model DP-systems. In [26], timed-arc Petri nets were consistently combined with 'nets-within-nets' hierarchical structure and sound time semantics was provided; also, the decidability of behavioural coverability-related properties using the notion of well-structured transition systems was established. For recent review of results on TAPN-nets and their verification, we refer to [27]. Moreover, TaPN-nets have comprehensive tool support via TAPAAL and UPPAAL software. Tool UPPAAL makes it possible to model and verify networks of timed automata. TAPAAL is a tool for modelling, simulation, and verification of TaPN-nets; it can utilize UPPAAL as a verification engine.

While a lot of results on temporal properties of timed and untimed versions of Petri nets were obtained, quantitative statistics of the behaviour of Petri nets and their extensions are studied to much lesser extent. The translation of TaPN-nets into DP-systems can be used to overapproximate the number of different age-values in the TAPN-net. This translation is a part of the tool implemented within the frame of this works – it extracts a metric graph in GraphML format from a TaPN-net allowing one to use existing software for metric graphs to overapproximate the stabilization time or growth rate of the number of clocks in the TaPN-net.

In Section 2, basic notions and notations are given. Section 3 covers the translation between DP-systems and TaPN-nets and implementation details. In Section 4, we demonstrate how to verify time-temporal properties of a DP-system using the suggested approach; results of performance experiments are provided. Section 5 concludes the paper with some discussion.

2. Preliminaries

By $\mathrm{N}, \mathrm{Q}_{\geq 0}, \mathrm{R}$, we denote the sets of natural, non-negative rational, and real numbers, respectively. The set of open and closed intervals over $\mathrm{Q}_{\geq 0} \cup \{\infty\}$ is denoted by $\mathrm{I}(\mathrm{Q}_{\geq 0})$. For a set S, a *bag* (*multiset*) m over S is a mapping $m : S \to \mathrm{N}$. The set of all bags over S is denoted by 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 comparisons of bags by $=, <, >, \leq, \geq$. We start by giving the definition of a metric graph [3].

Definition 1 (Metric graph). *A graph Γ is said to be a metric graph, if*

- *each arc a is assigned a positive length $l(a) \in (0, \infty)$. An arc with $l(a) = \infty$ has only one incident vertex and is called a lead;*
- *the lengths of the arcs that are reversals of each other are assumed to be equal;*
- *a coordinate $x(a) \in [0, l(a)]$ increasing in the direction of the arc is assigned on each arc;*

- *the relation* $x(a') = l(a) - x(a)$ *holds between the coordinates on mutually reversed arcs.*
- *for arc* a*, the length* $l(e)$ *of its support edge* e *is equal to* $l(a)$.

A state s of a *DP*-system D on a metric graph Γ is a set of points located on the arcs of Γ. When time starts to flow, each point p moves along its arc a direction from 0 to $l(a)$ with the same velocity; its position is denoted by $x(p)$. If p reaches a vertex v, new points are issued to all the outgoing arcs of v (intuitively, this corresponds to wave packet scattering). New points generation may result in more than one point on some arcs. Each produced point p' starts moving along corresponding e. 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. However, points do not collide anywhere on edges except vertices, i.e., if two points met on an edge, they both continue their movement towards their own directions. In fig. 1, the initial set of points consists of two points in vertices v_1 and v_3. The point in v_1 produces a new point on edge $\{v_1, v_2\}$, The point in 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 gray), but there are points (coloured black) moving from v_2 and v_3 to their adjacent vertices.

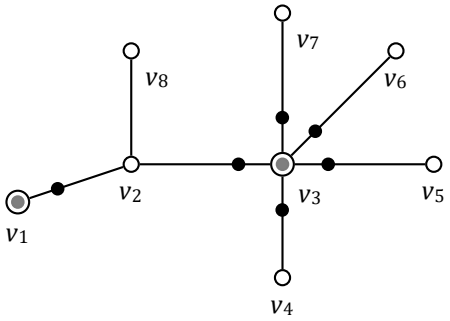

Fig.1. System of dynamic points D on metric graph Γ

The stabilization time $t_s(\Gamma)$ of a metric graph Γ with commensurable edge lengths is the value of the period of time from the starting point to the point in time when the number of points $N_\Gamma(t)$ on the graph has been stabilized [6], *i.e.*, $\forall t \geq t_s(\Gamma) : N_\Gamma(t) = N_\Gamma(t_s(\Gamma))$. Henceforth, when stabilization time is discussed, discrete time instants when points collapse at vertices are excluded from consideration as, strictly, at these instants the number of points can decrease.

A place/transition net (PT-net) is a Petri net with indistinguishable tokens.

Definition 2 (Place/transition nets). A PT-net is a tuple $< P, T, F, \gamma >$, 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 $< y, x >$ is called an *input arc*, and arc $< x, y >$ – an *output arc*. A *preset* $\bullet x$ and a *postset* $x \bullet$ are subsets of $P \cup T$ such that $\bullet x = \{y \mid < y, x > \in F\}$ and $x \bullet = \{y \mid < x, y > \in F\}$. A *marking* of N is a function $m: P \to$ N. A pair $< N, m >$ of a PT-net and a marking is called a marked net.

Let $N = < P, T, F, \gamma >$ be a PT-net. A transition $t \in T$ is *enabled* in a marking m iff $\forall p \in \bullet t \Rightarrow m(p) \geq \gamma < p, t >$. An enabled transition t can *fire* yielding a new marking $m'(p) = m(p) - \gamma < p, t > + \gamma < t, p >$ for each $p \in P$ (denoted $m \to^t m'$).

Now, we provide the definition of Timed-Arc Petri nets with token-based time semantics and urgency [2], [12], [24].

Definition 3 (Timed-Arc Petri net with Urgency). A TaPN-net is a tuple $TaPN = < N, \gamma^t, U >$, where

- $N = < P, T, F, \gamma >$ *is a PT-net called the skeleton of TaPN and denoted by* $S(TaPN)$*;*
- $\gamma^t: P \times T \rightarrow I(Q \geq 0)$ *is a set of token-age constraints on arcs;*
- $U : T \rightarrow \mathrm{Q}_{\geq 0}$ *is a set of urgency constraints on transitions.*

The marking $m = < m_s, m_t, m_u >$ of a TaPN-net $TaPN$ consists of a marking m_s of $S(TaPN)$, a time marking $m_t : Tok(m_s) \rightarrow \mathrm{Q}_{\geq 0}$ that assigns clocks to tokens, and an urgency marking $m_u : T(m_s) \rightarrow \mathrm{Q}_{\geq 0}$ that assigns clocks to transitions, where $T(m_s)$ comprises all transitions and $Tok(m_s)$ comprises all tokens of the marked PT-net $< S(TaPN), m_s >$. 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 $\gamma^t(p,t)$ means that t may fire only by consuming a 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. A transition t is enabled in the marking $m = < m_s, m_t, m_u >$, if t is enabled in $< S(TaPN), m_s >$ and time constraints of t are satisfied, *i.e.*, each token z from a place p involved in the firing of t satisfies $m_t(z) \in \gamma^t(p,t)$.

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 $z \in m_s : (m_t + \delta)(z) = m_t(z) + \delta$, 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. Automated translation of DP-systems into TaPN-nets

In the context of communication networks and computer systems, we may consider a distributed system of communicating reactive sensor-nodes (or other computational devices). One of the nodes is a server that initiates communication by sending control messages (signals) while other sensor-nodes react on such signals. On the assumption that the processing speed of nodes is much higher than the speed of signal propagation, when a node receives one or more signals from its neighbour nodes, it responds instantly by sending signals to some of its neighbours. In an extreme case, each node sends signals to all of its neighbours upon each message arrival. This extreme behaviour corresponds to the dynamics of a DP-system enabling us to use metric graphs to overapproximate messages scattering in networks.

In the latter case, time-temporal properties of DP-systems become a matter of interest [28]. However, up to the knowledge of authors, there are no tools that enable us to conduct analysis of such natural time-temporal properties as:

- Is it possible that more than N signals come to node X in K seconds;
- Is it possible that in the next K minutes two messages come with the difference of their time moments less that ε milliseconds;
- For the system initial phase of duration N, if a point comes to node X at time T, then no points comes to node Y within K seconds from T;

In this section, we provide an algorithm for the translation of DP-systems in TaPN-nets. Tool TAPAAL supports verification of specifications in timed computation tree logic (TCTL) language, which allows expressing time and temporal properties of process dynamics [29] [30]. In the next section, we demonstrate how to express these properties in TCTL-properties of converted TaPN-nets.

To simulate *DP*-systems with models of Petri nets with real-time semantics, we need to represent points moving along edges 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 is not fixed and may grow (infinitely when edge lengths are incommensurable); therefore, it is not possible to model *DP*-system using clocks in time or timed Petri nets [19], [20] as these models have finite structurally-determined number of clocks. In *TaPN*-nets, clocks are assigned to tokens [12], and the number of clocks can grow along with the number of tokens in the net.

We start with the description of *DP*-system to *TaPN*-net translation procedure, and, later, cover some design and technical details on the implementation of the translation.

We denote the set of points on the edge e by $P(e)$. For a lead $< v_1, v_2 >$, one of v_1 and v_2 is marked with infinity. Note that, as movement of a point along an edge corresponds to a continuous process, we model edges of DP-system with TaPN-net places; and, as arrival of a point at a vertex corresponds to a discrete event, we model vertices of DP-system with TaPN-net transitions.

3.1 Algorithm for translation from a DP-system to a TAPN-net procedure

Step 1. For each edge e connecting vertices a and b in D, we create places e_{ab} and e_{ba} in N_{TA};

Step 2. For each lead $< a, b >$ in D with b marked with infinity, an arc from vertex a to a distinct infinity-vertex is introduced in N_{TA}. We need to handle points moving away from vertex a separately.

(a) create place e_{ab}, transition t_{ab} with urgency 0, and an arc from e_{ab} to t_{ab} with time interval [0,0];

(b) for each point on the arc $< a, b >$

i. create a token with 0 time at place e_{ab};

ii. create transition t_{ba} with urgency 0;

(c) for each point p_i on arc $< b, a >$

i. create place e_{ba}^{i} with a token;

ii. create an arc from e_{ba}^{i} to t_{ba} with time interval $[x(p_i), x(p_i)]$;

(d) for each arc from a to c, where $c \neq b$, create an arc from t_{ba} to e_{ac};

Step 3. For each already created place e_{ab} in N_{TA} such that vertex a is not marked with infinity in D

(a) for each point p on arc $< a, b >$

i. create token at e_{ab} with timer set to $x(p)$;

(b) create transition t_{ab} with 0 urgency if b is not marked with infinity;

(c) create an arc with time interval $[l(e), l(e)]$ from e_{ab} to t_{ab};

(d) for each arc $< b, c >$ in D, create an arc from t_{ab} to e_{bc} in N_{TA};

(e) create transition t'_{ab}, an arc from e_{ab} to t'_{ab} with multiplicity 2, urgency 0, and zero time interval, and arc from t'_{ab} back to e_{ab}.

An example of such conversion is illustrated in fig. 2. As each undirected edge in D corresponds to two opposite arcs, the number of places in NTA on the right is twice as the number of vertices in D on the left. Transitions corresponds to the event of a point reaching a vertex; auxiliary transitions are introduced to clean redundant tokens from edge-places in NTA to handle events when multiple point came to a vertex simultaneously.

3.2 Algorithm for translation from timed-arc Petri nets to metric graphs

To use the results on the asympotics and estimates on growth and stabilization time of the number of dynamic points in *DP*-system [6] to overapproximate *TaPN*-net the number of different token age-values, we implement the translation of a restricted class of *TaPN*-nets into *DP*-systems.

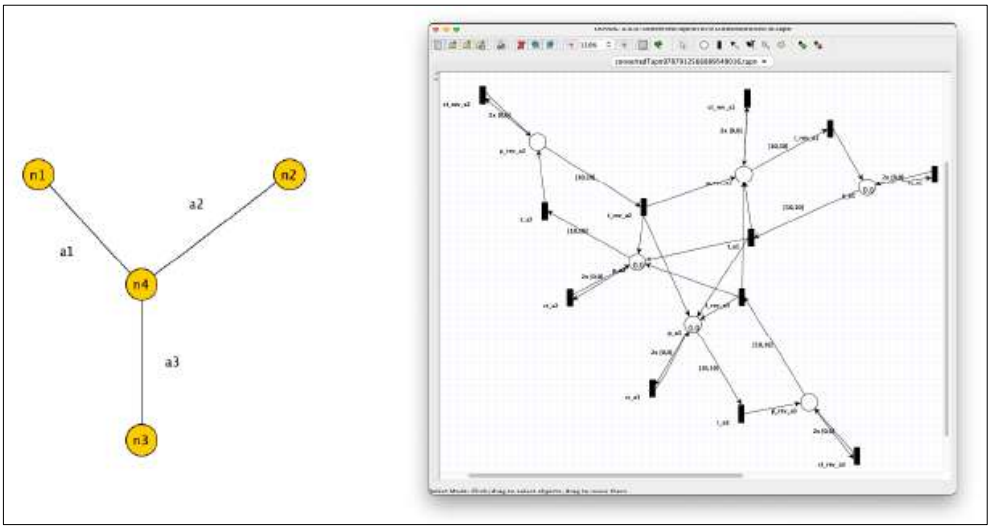

Fig.2. Metric graph D on the left and the resulting TAPN N_{TA} on the right

Let us consider a class of TaPN-nets under the following restrictions:

1) The skeleton of $TaPN$-net is a marked graph, i.e., every place has exactly one incoming arc, and exactly one outgoing arc [1];
2) All urgency time specifications are 0;
3) All arcs are labelled with point-intervals only, i.e., intervals of form $[t, t]$.

Let NTA be an input TaPN-net and D be the resulting DP-system. The next procedure converts the TaPN-net to a DP-system. The DP-system D generates point each time NTA produces a new token with timer.

1) For each transition t_i in N_{TA}, create vertex v_i in D;
2) For each transition t_i, place p, and transition t_j, such that there is an arc from t_i to p and an arc from p to t_j
   a) create an edge connecting vertices v_i and v_j (which correspond to transitions t_i and t_j, respectively) with a length equal to the time interval on the arc from p to t_i; this procedure may introduce multiple edges or self-loops, thus, an intermediate graph may be not a metric graph; we provide a resolving strategy after the procedure;
   b) for each token in p, put a point on beginning of edge from v_i to v_j, *i.e.,* at vertex v_i;
3) For each place p, which has only inbound arcs or only outbound arcs
   a) for each adjacent transition t, create a lead from corresponding vertex v;
   b) if p has only outbound arcs, then for each token in p, for each adjacent transition t_i, put a point on the lead incident to corresponding vertex v_i moving towards v_i and located on the distance equal to the point-interval on arc $< p, t_i >$.

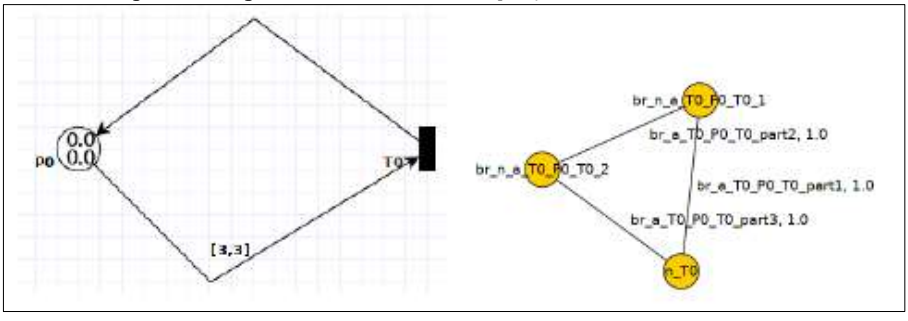

Fig.3. Example of self loop break

To resolve multiple edges and self-loops introduced during the translation, we break one of edges e into two edges by introduce a new vertex; lengths of these new edges are equal to the half $l(e)$. For self loops, we need to introduce two vertices, and they breaks edge e into three parts with lengths equal to a third of $l(e)$. This step, firstly, breaks the loop with one edge, and, secondly, breaks multiple edges; the result of such step for a loop transition is illustrated in fig. 3.

3.3 Implementation details

The translation is implemented in tool **mg-to-tapn** as a TAPAAL extension; the tool uses TAPAAL internal representation for storing and analysis of TaPN-nets. A JSON-based file format for processing DP-systems was developed and its support was implemented. The file format is an extension of JSON Graph Format [31]. To allow visualization of DP-systems, a support for GraphML-based format compatible with yEd Graph Editor [32] was implemented.
As **mg-to-tapn** is an extension of TAPAAL, the following functions can be utilized both through command line interface or graphical user interface of TAPAAL.

- Translate DP-system in JSON format to TaPN-net in TAPAAL representation;
- Translate TaPN-net in TAPAAL representation to DP-system in JSON format;
- Translate TaPN-net in TAPAAL representation to DP-system in GraphML-based yEd-compatible format.

4. Checking TCTL properties of DP-system and experimental results

In this section, we demonstrate how to check DP-system time-temporal properties of a metric graph using the implemented translation and provide results of performance experiments.

4.1 Checking time-temporal properties of a metric graph

Tool *TAPAAL* does not directly support the whole class of *TCTL* properties; therefore, for time-temporal properties, we need to combine supported verification engines with additional auxiliary net fragments to capture such properties. We demonstrate how to encode the following propertiesusing this technique.

i. **Is it possible that more than N signals come to node X in K seconds**. Firstly, we need to capture the property that more than N signals come to node X. Arrival of a signal to X from Y along arc $< Y, X >$ corresponds to a firing of transition t_{YX} in *TaPN* (see naming strategy in section 3). We add a place p'_X, which will be used as a counter for the number of signals-tokens that came to X. Then, we connect each t_{YX} to p'_X with an arc; each firing of t_{YX} will produce a token in p'_X. We call such a place *counter for* X. In addition, we need to determine a time point when K seconds has passed. We introduce a new place p_{start} and put a token in p_{start}. We add urgent transition t_{time} and connect p_{start} and $t_{time}$with an arc specified by time interval $[K, K]$. So, place p_{start} becomes empty when K seconds has passed. We call such a net fragment consisting of place p_{start} with a token, transition t_{time}, and arc $< p_{start}, t_{time} >$ specified by $[K, K]$ as *timer for* K.

   Finally, we can express the property as whether there is a reachable state when p'_X has K or more tokens, and p_{start} has exactly one token. This property is purely temporal while time features are captured using time specifications in *timer for K*. In CTL, the property is expressed as $EF((m(p'_X) \geq N)\ \&\ (m(p_{start}) = 1))$.

ii. **Is it possible that in the next K minutes two messages come with the difference of their time moments less that ε milliseconds**. To check the property, we add *timer for K* construction, *counter for* X, and transition t'. We add arc $< p'_X,\ t' >$ labelled with $[\varepsilon, \varepsilon]$. To make , t' urgent, we use age invariant $Inv: \leq \varepsilon$ on p'_X. If there are two or more tokens in p'_X, , then those tokens come to node X with time difference less than ε. The corresponding CTL property is $EF\ ((p'_X \geq 2)\ \&\ (p_{start} = 1))$.

iii. **For the system initial phase of duration N, if a point comes to node X at time T, then no points comes to node Y within K seconds from T.** We add *counter for* X, *counter for* Y, and *timer for* N. We add transition t'_X, arc $< p'_X, t'_X >$labelled with $[K, K]$, age invariant $Inv: \leq K'$ on p'_X making t'_X urgent, urgent transition t'_Y, and arc $< p'_Y, t'_Y >$. If a token is in p'_X for K seconds, then t'_X fires consuming it; if a token came to p'_Y, it leaves the system via t'_Y immediately.

The property become violated if p'_X and p'_Y hold tokens simultaneously, as it corresponds to a state where a token, which timer value is less than K, is in X, and a token came to Y. The resulting CTL property is $AG\neg((p'_X \geq 1)\ \&\ (p'_Y \geq 1)\ \&\ (p_{start} = 1))$.

Table 1. Performance results for verifying the first property

Graph type/ size (in nodes)	**3**	**5**	**10**	**20**	**30**
Complete	0.05s	0.11s	1.36s (11mb)	41.29s (26mb)	–
Key	0.03s	0.07s	0.09s	0.4s (6mb)	11.91s (86mb)
Cycle	0.02s	0.03s	0.03s	0.02s	0.03s
Star	0.05s	0.09s	1.10s (13mb)	51.53s (353mb)	> 300s (> 1.8gb)

Table 2. Performance results for verifying the third property

Graph type/ size (in nodes)	**3**	**5**	**10**	**20**	**30**
Complete	0.01s	0.01s	0.03s	0.65s (11mb)	–
Key	0.01s	0.01s	0.08s	4.21s (45mb)	> 300s (> 1.8gb)
Cycle	0.01s	0.01s	0.01s	0.01s	0.06s / 2mb
Star	0.01s	0.01s	> 300s (> 1.6gb)	14.16s (120mb)	> 300s (> 1.1gb)

4.2 Experimental results

We conducted experiments on metric graphs of different topology: star graph, complete graph, cycle graph, and «key» graph (linear graph joined with a cycle of 3 elements on the end). The metric graphs were converted to *TaPN*-nets in *TAPAAL* format. The first and third property, encoded as suggested in the previous section, were checked for the converted metric graphs; verification time and memory consumed were measured.

The results for the first property are given in Table 1. For this experiment, we checked the first property for the initial fragment of 1000 time units. Time and consumed memory were measured; if the amount of consumed memory was negligible, the value is not provided. The results for metric graph K_{30} were not obtained as more than a thousand nodes in the resulting *TaPN*-net exceeded some internal restrictions of *TAPAAL*. The low verification time for a cycle metric graph is stipulated by the low node degrees in the graph thus leading to a smaller number of transitions in the converted *TaPN*-net. The results for the third property are given in Table 2. For this experiment, we checked the second property with $K = 20$ and $N = 1000$. We see almost the same picture with the exception of that for key metric graphs the memory restrictions come in play earlier. The experiments were performed on a personal computer with 2 GHz Quad-Core Intel Core i5 processor and 16 GB of memory (3733 MHz LPDDR4X) running under macOS 11.0.1 operating system.

5. Conclusion

In this paper, we developed an approach that enable us to analyze TCTL-properties of DP-systems based on the implementation of the translation between DP-systems and TaPN-nets. We demonstrated how to conduct analysis of TCTL properties of DP-systems by adding an auxiliary net

fragment. For several time-temporal properties of metric graphs, we showed how to express the properties in TAPAAL using an auxiliary subnet and TAPAAL TCTL queries.

The automated translation allows using the well-known efficient tool TAPAAL to carry out simulation and verification (analysis of TCTL properties) of DP-system dynamics and interpreting results for DP-systems on a given restricted subclass of TaPN-nets.

The experiments showed that for scarce mid-sized graphs analysis could be done quite effectively. As these experiments were conducted for the first time, to get more straightforward results, we neither used any optimizations nor tried to encode properties in a more compressed manner. Therefore, we expect that optimizations could considerably improve performance on the benchmark tests as complete and star graphs possess much symmetry.

To improve the precision of the overapproximation of a TaPN-net with a DP-system, the already known counting functions and asymptotics for undirected metric graphs shall be extended to directed metric graphs. This direction is being under current research.

References / Список литературы

[1]. Reisig W. Understanding Petri Nets – Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013, 257 p.

[2]. Hanisch H.-M. Analysis of place/transition nets with timed arcs and its application to batch process control. Lecture Notes in Computer Science, vol. 691, 1993, pp. 282–299.

[3]. Berkolaiko G. and Kuchment P. Introduction to quantum graphs. American Mathematical Society, 2013, 270 p

[4]. Berkolaiko G., Carlson R., Fulling S. A., Kuchment P., eds. Quantum Graphs and Their Applications. Proc. of an AMS-IMS-SIAM Joint Summer Research Conference on Quantum Graphs and Their Applications. American Mathematical Society, 2006, 307 p.

[5]. Чернышев В.Л. Нестационарное уравнение Шрёдингера:статистика распространения гауссовых пакетов на геометрическом графе. Труды Математического института им. В.А. Стеклова, том 270, 2010 г., стр. 249–265 / Chernyshev V. L. Time-dependent Schrödinger equation: Statistics of the distribution of Gaussian packets on a metric graph. Proceedings of the Steklov Institute of Mathematics, vol. 270, 2010, pp. 246–262.

[6]. Tolchennikov A.A., Chernyshev V.L., 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, no. 5, 2016, pp. 531–537.

[7]. Exner P. and Lipovsky J. Topological bulk-edge effects in quantum graph transport. Physics Letters A, vol. 384, issue 18, 2020, article id 126390.

[8]. 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, no. 3, 2010 г, стр. 623-638 A. A. Tolchennikov, V. L. Chernyshev, A. I. Shafarevich, Asymptotic properties and classical dynamical systems in quantum problems on singular spaces, Russian Journal of Nonlinear Dynamics, vol. 6, no. 3, 2010, pp. 623–638 (in Russian).

[9]. Chernyshev V.L., Tolchennikov A.A. Asymptotic estimate for the counting problems corresponding to the dynamical system on some decorated graphs. Ergodic Theory and Dynamical Systems, vol. 38, no. 5, 2018, pp. 1697–1708.

[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, no. 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, no. 8, 2017, pp. 937–948.

[12]. Bolognesi T., Lucidi F., and 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. 395–408.

[13]. David A., Jacobsen L., Jacobsen M., Jørgensen K., Møller M., and Srba J. TAPAAL 2.0: integrated development environment for timed-arc Petri nets. Lecture Notes in Computer Science, vol. 7214, 2012, pp. 492–497.

[14]. Model checking contest 2020. Available at: https://mcc.lip6.fr/models.php, accessed June 2020.

[15]. Esparza J. Decidability and complexity of Petri net problems – an introduction. Lecture Notes in Computer Science, vol. 1491, 1998, pp. 374–428.
[16]. Czerwinski W., Lasota S., Lazic R., Leroux J., Mazowiecki F. The reachability problem for Petri nets is not elementary. In Proc. of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019, pp. 24–33.
[17]. Berard B., Cassez F., Haddad S., Lime D., and Roux O. H. Comparison of different semantics for time Petri nets. Lecture Notes in Computer Science, vol. 3703, 2005, pp. 293–307.
[18]. Brown C. and Gurr D. Timing Petri nets categorically. Lecture Notes in Computer Science, vol. 623, 1992, pp. 571–582.
[19]. Merlin P. A study of the recoverability of computer systems. Ph. D. Thesis, Dept. of Information and Computer Science, University of California, Irvine, CA, 1974.
[20]. Ramchandani C. Analysis of asynchronous concurrent systems by timed Petri nets. Ph. D. Thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, 1974.
[21]. 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.
[22]. 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 / 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.
[23]. Vinarskii E.M., Zakharov V.A. On the verification of strictly deterministic behaviour of Timed Finite State Machines. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 3, 2018, pp. 325-340. DOI: 10.15514/ISPRAS-2018-30(3)-22.
[24]. Akshay S., Genest B., and Helouet L. Decidable Classes of Unbounded Petri Nets with Time and Urgency. In Proc. of the 37th International Conference on Application and Theory of Petri Nets and Concurrency, 2016, pp. 301–322.
[25]. Dworzanski L.W. Bounding untimed Petri net using timing operation. Report on doctoral intermediate thesis results, 2013-09-16. Available at http://mathtech.ru/dworzanski/talks/bounding.html
[26]. Dworzanski L.W. Consistent timed semantics for nested Petri nets with restricted urgency. Lecture Notes in Computer Science, vol. 9884, 2016, pp. 3-18.
[27]. Jacobsen L., Jacobsen M., Møller M. H., Srba J. Verification of timed-arc Petri nets. Lecture Notes in Computer Science, vol.6543, 2011, pp. 46-72.
[28]. Shoshmina I.V. Developing formal temporal requirements to distributed program systems. System informatics, no. 8, 2016, pp. 21-32.
[29]. Alur R, Courcoubetis C, Dill D. Model-checking in dense real-time. Information and Computation, vol. 104, no. 1, 1993, pp. 2-34.
[30]. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. БХВ-Петербург, 2010 г., 552 стр. / Karpov Yu. G., Model checking. Verification of parallel and distributed systems. BHV-Peterburg, 2010, 552 p. (in Russian).
[31]. JSON Graph Format (JGF). Available at: http://jsongraphformat.info, accessed November 2020.
[32]. yEd Graph Editor: High-quality diagrams made easy. Available at: https://www.yworks.com/products/yed, accessed November 2020.

Information about authors / Информация об авторах

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.

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.

Александр Александрович ИЗМАЙЛОВ проходит обучение в департаменте программной инженерии факультет компьютерных наук НИУ ВШЭ.

Aleksandr Aleksandrovich IZMAYLOV studies at School of Software Engineering Faculty of Computer Science National Research University Higher School of Economics.

DOI: 10.15514/ISPRAS–2020–32(6)–13

Моделирование технических и математических задач прикладных областей знаний на ЭВМ

[1,2] *Е.М. Лаврищева, ORCID: 0000-0002-1160-1077 <lavr@ispras.ru>*
[2] *И.Б.Петров, ORCID: 0000-0003-3978-9072 <petrov@mipt.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Московский физико-технический институт,*
141701, Россия, Московская обл., г. Долгопрудный, Институтский пер., д. 9

Аннотация. Рассмотрено моделирование технических задач и задач прикладной математики, их алгоритмизация и программирование. Дается характеристика численного моделирования технических задач и прикладной математики: физико-технических экспериментов, энергетических, баллистических и сейсмических методов И.В Курчатова, начиная с математических методов 17-20 вв., первых ЭВМ и компьютеров. Дан анализ первых технических задач и задач прикладной математики, их моделирование, алгоритмизация и программирование с помощью граф-схемного языка А.А.Ляпунова, адресного языка и языков программирования. Представлены численные методы, реализованные под руководством А.А. Дородницына, А.А.Самарского, О.М. Белоцерковского и других ученых на современных суперкомпьютерах. Приведены примеры математического моделирования биологической задачи лечения глаз и предмета «Вычислительной геометрии» в Интернет.

Ключевые слова: математика; модель; моделирование; алгоритм; граф; области знаний; биология; геометрия

Для цитирования: Лаврищева Е.М., Петров И.Б. Моделирование технических средств и задач прикладной математики на ЭВМ. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 167-182. DOI: 10.15514/ISPRAS–2020–32(6)–13

Благодарности: Работа поддержана грантом РФФИ № 19-01-00206.

Modeling Technical and Mathematical Tasks of Applied Knowledge Areas on Computers

[1,2] *E.M. Lavrischeva, ORCID: 0000-0002-1160-1077 < lavr@ispras.ru>*
[2] *I.B.Petrov, ORCID: 0000-0003-3978-9072 <petrov@mipt.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia,
[2] *Moscow Institute of Physics and Technology (MIPT)*
141700, Russia, Moscow region, Dolgoprudny, Campus per., 9

Abstract. The paper considers modeling of technical problems and problems of applied mathematics, their algorithms and programming. The characteristics of the numerical modeling of technical problems and applied mathematics are given: physical and technical experiments, energy, ballistic and seismic methods of I.V. Kurchatov, starting with mathematical methods of the 17-20th centuries, the first computers and computers. The analysis of the first technical problems and problems of applied mathematics, their modeling, algorithmization and programming using the A.A. Lyapunov graph-schematic language, address language and programming languages is given. Numerical methods are presented, implemented under the guidance of A.A. Dorodnitsyn, A.A. Samarsky, O.M. Belotserkovsky and other scientists on modern supercomputers. Examples

of mathematical modeling of the biological problem of eye treatment and the subject of «Computational geometry» on the Internet are given.

Keywords: mathematics; model; modeling; algorithm; graph; knowledge areas; biology; geometry.

For citation: Lavrishcheva E.M., Petrov I.B. Modeling of technical means and problems of applied mathematics on computers. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 167-182 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–13

Acknowledgements. This work was supported by the RFBR grant No. 19-01-00206.

1. Введение. Становление моделирования техники и математики

1.1 Моделирование технических средств

Моделирование в технике используется уже сотни лет в строительстве зданий, мостов, кораблей и др. Техническая модель отражала структурную сущность внешности технического предмета или явления и связей отдельных его элементов.

Значительный подъем исследований по прикладной математике в СССР произошел после окончания ВОВ 1945 года по следующим направлениям развития:

- передовые системы вооружений, обусловленные начавшейся гонкой вооружений между СССР и США;
- ядерные программы, требовавшие проведения многочисленных численных расчетных работ;
- ракетостроение, требующее проведение сложных аэродинамических, баллистических и прочностных численных расчетов;
- электронная техника, системы радиосвязи;
- физические методы разведки нефти и газа и методы их добычи в нашей стране.

Создание передовых систем вооружений началось в рамках ВПК, начиная с 1945 г. Министерство радиопромышленности СССР создало программу развития радиотехнических и радиофизических приборов с обеспечением высокого качества и безопасности. В рамках ВПК создавались специализированные приборы и устройства радиотехнического, радиолокационного, авиационного, космического назначения. Так, в 1960 году в космос полетел Ю.А. Гагарин, и весь мир был поражен уровню достижений нашей науки и техники.

В этот период (МНИИПА, Липаев В.В.) было создано более 100 средств вычислительной техники (ВТ), специализированные ЭВМ (ПРА-6.0, МАПА, АРГОН, АОУ6 и др.), а также радиолокационная техника для наведения и слежения за движением военных самолетов, подводных лодок, кораблей и др. Кроме того, под руководством Липаева В.В. были созданы программные комплексы ПРОТВА, ЯУЗА, РУЗА, ПРОМЕТЕЙ для реализации военных задач [1].

1.2. Моделирование задач прикладной математики

Первые сведения по прикладной математике относятся к XVII в. Развитие небесной механики и геодезии в связи с потребностями навигации и мореплавания, составление таблиц тригонометрических функций, появление артиллерии диктовали необходимость разработок расчетных методов. В это время появляется важнейший математический аппарат для решения многих прикладных задач – интегральное и дифференциальное исчисление, разработанное И. Ньютоном и Г. Лейбницем. Для решения дифференциальных и интегральных уравнений были разработаны приближенные, численные методы Эйлера решения задачи Коши обыкновенных дифференциальных уравнений и численного интегрирования. В связи с развитием небесной механики (Лапласом) и механики сплошных сред появилась методы решения уравнений в частных производных и численного решения

систем линейных и нелинейных алгебраических уравнений (Ньютон, Якоби, Гаусс–Зейдель, Лежандр, Эрмит и др.).

Важную роль в развитии теории приближенных вычислений – одной из основных теорий и в функциональном анализе, и в вычислительной математике, сыграли работы П.Л. Чебышева (чебышевская система функций, чебышевские многочлены, теория равномерных приближений и др.). В начале прошлого столетия получили развитие численные методы решения обыкновенных дифференциальных уравнений (работы Галеркина, Ритца, Рунге, Крылова, Кутты, Розенберга, Ван дер Поля, Адамса, Бутчера и др.), что позволило получить численные решения многих прикладных задач.

В XIX в. в прикладной математике появились нелинейные разностные отображения, заложившие основы теории нелинейных процессов. В начале XX в. возникала теория уравнений Максвелла, описывающая электрические и электромагнитные поля во времени и пространстве, а также появились методы решения систем уравнений механики сплошных сред. Существовавшие в то время простейшие численные приборы и арифмометры позволяли решать математические задачи, задавая цепочки последовательных расчетных алгоритмических действий в соответствии с алгоритмом Аль-Хорезми. Численные расчеты проводились на аналитических машинах.

2. Методы моделирования и программирования технических и математических задач на ЭВМ

2.1. Моделирование математических задач энергетики, баллистики и геофизики

Ядерные бомбардировки Хирасимо и Нагасаки в августе 1945 года послужили стимулом для ускоренного создания в нашей стране под руководством И.В. Курчатова атомной энергетики и проведения многочисленных численных расчетов, связанных с распадом ядерного урана и плутония, для ядерной бомбы. Была создана теория цепной реакции распада урана, и сформировались математические методы ядерной энергетики. Это привело к созданию в стране новых энергетических и технических устройств и атомной бомбы в 1949 году. Кроме того, были создана первые в мире АЭС (1954 г.) и атомный ледокол «Ленин» (1957 г.).

Моделирование теоретических методов в энергетике и баллистике активно начали проводить в связи с появления первого отечественного компьютера МЭСМ, созданного коллективом лаборатории Института электротехники АН УССР под руководством С.А. Лебедева (1948-1951). На МЭСМ под руководством академика М.В. Келдыша реализовывались математические задачи внешней баллистика, электрические и технические задачи:

- составление таблиц для статистического приемочного контроля задач баллистики;
- динамика в теории упругости;
- обеспечения устойчивости энергосистем (Куйбышевская ГЭС);
- расчета тепловых напряжений строительных конструкций;
- выбор оптимальных параметров шахтных канатов;
- геодезические задачи, оценка объемов земляных работ автодорог (и БАМ).

Алгоритмы таких задач разрабатывались путем декомпозиции математической задачи на отдельные элементарные функции и их описание средствами математическая логики, граф-схемного языка А.А. Ляпунова или адресного языка В.С. Королюка и Е.Л. Ющенко (1955). Алгоритмы некоторых математических задач представлялись в виде стандартных математических подпрограмм в Библиотеках программ первых ЭВМ. Например, метод Адамса-Штермера/Дирихле был реализован В.М. Глушковым путем декомпозиции на отдельные элементарные математические операции, отработан на МЭСМ и оформлен в виде

стандартной подпрограммы [2]. Первая система загрузки для библиотек стандартных математических подпрограмм ИС-2 для М20 была создана в ИПМ [3].

Моделирование задач прикладной математики для решения на первых ЭВМ проводилось с использованием теории граф-схемного языка А.А. Ляпунова (1955). В этом языке делалось описание математических элементов задачи с помощью математических операций, формальных логических символов и операторов по ориентированному графу переходов с одной входной и одной выходной вершиной [4, стр. 226-253]. Граф задавал структуру алгоритма, в которой одна вершина – преобразователь и две вершины – распознаватели. Две граф-схемные программы функционально эквивалентны, если для любой интерпретации соответствующие программы вычисляют одинаковые функции. Каждому выполнению программы сопоставляется протокол порядка следования базовых операций. Если известны значения истинности предикатов, входящих в программу, то протокол строится однозначно и не требует интерпретации операций. Если выбор значения истинности предикатов произвольный, то для схемы программы создается некоторое множество протоколов, называемое *детерминантом.* С учетом этого понятия, доказывается эквивалентность двух граф-схем, если их детерминанты совпадают. Формальная эквивалентность корректна, если из нее следует функциональная эквивалентность. Детерминант описывается формальным языком, который воспринимается некоторым конечным автоматом. При этом формальная эквивалентность разрешима и совпадает с функциональной. Существенное место в теории граф-схемного программирования занимает перевод граф-схем из одной модели в другую. Проблемы разрешимости соответствующей формальной эквивалентности развивались в работах Р.И. Подловченко и С.С. Лаврова [5].

2.2. Моделирование математических задач сохранения энергии, зарядов и частиц

Основы вычислительных методов решения уравнений в частных производных были определены математиками А.Н, Тихоновым, А.А. Самарским, В.С. Рябеньким, американскими учеными Р. Курантом, П. Лаксом и др. Были введены фундаментальные понятия теории разностных схем: сходимость, аппроксимация, устойчивость, доказана базовая теорема эквивалентности. Под руководством академика А.А.Самарского проводилось моделирование задач прикладной математики согласно сформировавшейся концептуальной модели предметной области знаний: КМ $= < X, R, F >$, где

X – конечное множество понятий предметной области;

R – конечное множество сущностных отношений (связей) между понятиями предметной области;

F – конечное множество функциональных алгоритмов задач предметной области.

КМ-модель задает информацию о понятиях предметной области в виде сущностных понятий, которые сохраняются в тезаурусе с возможными связями между ними. Функциональная (техническая) задача формулируются в рамках теории предметной области, в которой математическая модель сочетается с общематематическим знанием, задающим эту модель. Алгоритмизация модели состоит в отображении точного знания о задаче и операциях над ее элементами в описании алгоритма программы для компьютера.

Модели программ разрабатывались для решения математических, физических и технических задач вначале на аналитических машинах и потом на ЭВМ [6]. Были разработаны математические модели на основе закона сохранения энергии, импульсов, зарядов и частиц, уравнений механики сплошных сред, электродинамики Максвелла, уравнений Больцмана для разреженного газа, кванто-механического уравнения Шредингера и др.

Появились и другие математические задачи в классах климатических, космических, геофизических, сейсмических, термодинамических, радиолокационных, акустических медицинских, биологических, химических и физических задач. Каждый из указанных

разделов науки имеет «численную» часть, поэтому появились науки – вычислительная математика, численные методы, высокопроизводительные вычисления, информационные и интеллектуальные средства [7, 8].

2.3. Моделирование физических задач и механики сплошных сред

Проведение вычислительного эксперимента О.М.Белоцерковским и С.К. Годуновым проводились с помощью математической модели физического эксперимента, включающей характерные свойства физического явления, алгоритма свойств и характеристик этого явления в виде простых формул и операций программы численного решения задач (математической физики, нелинейной механики и др.) на ЭВМ. Одной из первых вычислительных задач была задача о ядерном взрыве и об обтекании затупленных тел потоком сверхзвукового газа. Исследование физических объектов проводилось путем анализа и выбора схемы эксперимента, определение элементов конструкции устройства. Затем составлялась разностная схема и проверка ее устойчивости. В результате получалось экспериментальное средство, которое отрабатывалось на серии экспериментов и проверки их правильности. Глубокое проникновение в механику сплошных сред привело к использованию интегральных и дифференциальных уравнений для описания происходящих явлений.

При математическом моделировании задач по этим уравнениям создавалась приближенная модель, которая отражала характерные свойства данного физического явления. С помощью компьютеров проводилось моделирование сложных технических и природных систем [9].

Разработка рациональных численных моделей вычислений привела к применению интеллектуальных гибридных *экспертных систем*, позволяющих прогнозировать поведение объектов и процессов. Экспертные системы являются инструментом распознавания образов и проведения систематических исследований вычислительных средств и формальных средств описаний ситуаций объектов. Система *распознавания* образов позволяет определить агрегированную оценку ситуаций, исходя из их формального описания. При установлении соответствия между классами эквивалентности, заданной на множестве решений и объектов, процедура распознавания ситуаций становится элементом автоматизации процесса принятия решений. Минимальное отклонение процесса управления от требуемого может возникнуть из-за изменения внешних условий и поэтому требуется прогнозировать изменения внешней среды с учетом временных интервалов. Проблемы *интеллектуализации* экспертных систем переносится на решение задачи поиска динамических инвариантов для соответствующих классов эволюционных задач. При моделировании численных задач в механике сплошных сред реализованы методы дискретных вихрей, метод Монте-Карло, метод неопределенных коэффициентов стационарных методов и метода расщепления физических процессов с сильно изменяющимися границами области интегрирования [8, 10].

2.4. Теория графов для моделирования задач техники и математики

Первая работа по теории графов была опубликована Эйлером в 1736 г. в статье про строительство Кенигсбергских мостов. В течение почти ста лет эта теория почти не использовалась. Интерес к теории графов возродился около середины XIX в связи с развитием естественных наук (электрических сетей, моделей кристаллов, структур молекул и др.) и формальной математической логики. Кроме того, оказалось, что решения многих математических головоломок, интеллектуализация шахматных задач формулировались в терминах теории графов. Последние 50 лет ознаменовались новыми интенсивными разработками теории графов в атомной энергетике, нейронных. нанобиологических, генетических, геофизических, космических задачах, а также в предметных областях знаний: физический эксперимент Коллайдер, Ядерный реактор, компьютерные, информационные, телекоммуникационные, экономические и бизнес-задачи.

Теория графов в программировании начала развиваться в школе А.П. Ершова (Касьянов В.И, Иткин В.Э., Евстигнеев А.А. и др.) [4, 11, 12]. Эта теория активно развивается в компьютерных науках и системах при их моделировании в виде графовой теории моделирования прикладных областей знаний [13-17]. Математический аппарат инцидентности и смежности используется для доказательства графовых, нейрографовых структур и др. [13-21]. Графовые структуры используются при реализации задач в биологии, медицине, физических экспериментах коллайдера, ядерного реактора и др. В ИСП РАН графовая теория применялась для моделирования вариантов OS Linux [15], средств защиты сложных компьютерных систем от разрушений и обеспечении надежности и безопасности функционирования программных систем [13-18] и др.

3. Численные методы моделирования прикладных задач на современных ЭВМ

3.1. Численные методы решения сложных задач

В 60–80-х гг. XX в. получили развитие итерационные методы, методы решения уравнений в частных производных эллиптического типа, методы решения задач в механике сплошных сред (Р.П. Федоренко, В.П. Колгана, Ван Лира, А. Хартена (TVD-схемы), Б.Н. Четверушкина, А. С. Холодова, С. Ошера, Ч.-В. Шу, И.Б. Петрова [22-24]. Численные методы высокого порядка точности были предложены в работах В. В. Русанова, С. Бурштейна, У. Рида, Т. Хилла (разрывный метод Галеркина), Э.Ф. Торо, А.И. Толстых. При численном решении многих практически важных задач используются методы конечных элементов, основанных на вариационных методах Галеркина и Ритца, в консервативных схемах, в задачах газодинамики, физики плазмы, магнитной гидродинамики, теории упругости и др. [25]. Многие из приведенных методов разработаны в Институте прикладной математики РАН, в Институте вычислительной математики РАН, Вычислительном центре РАН и др.

3.3. Применение графических плат GPU для решения математических задач

Инструментарий CUDA использует графические платы GPU для решения задач, описываемых уравнениями математической физики. Такие платы получили название GPGPU (general purpose computing on graphics processing units). Применение GPGPU связано с тем, что графических платы создавались для визуализации, которая не зависит от работы вычислительных ядер. Одним из путей выхода из этой ситуации является использование гибридных (гетерогенных) узлов, в которых соединены процессоры общего назначения GPU и GPGPU. В этой комбинации процессоры общего назначения берут на себя выполнение сложных логических операций, а GPGPU ведут обработку большого количества однородных потоков информации. Как показывает современная практика, такие гибридные суперкомпьютеры обладают при пиковой производительности приблизительно на порядок меньшую стоимость и энергопотребление, чем системы одинаковой производительности на основе процессоров общего назначения.

В комбинации с процессорами общего назначения использование графических плат сталкивается с серьезными трудностями. Задача состоит в создания алгоритмов, требующих переработки больших массивов информации типа Big Data. В настоящее время особую важность приобретают такие направления, как аэромеханика летательных аппаратов, прочность и надежность аэрокосмической техники, проектирование композиционных покрытий [26], безопасность железных дорог, сейсмостойкость объектов атомной энергетики, глобальная сейсмика [27], задачи, связанные с освоением запасов нефти и газа в условиях Севера и Арктики (безопасность и устойчивость ледостойких платформ, нефтегазопроводов, ледоколов и судов ледового класса, миграция крупных ледовых

образований) [28]. Важнейшей проблемой, решаемой с помощью высокопроизводительных вычислительных систем, является сейсмо- и электроразведка углеводородов, особенно в шельфовых зонах российских северных морей (прямые и обратные задачи георазведки). Суперкомпьютерное моделирование также позволяет моделировать процессы, происходящие при операциях, травмах и других процессах в медицинской практике и в биологических объектах [29-30].

3.4. Программные методы решения физико-технических задач

Для численного решения сложных физических задач, например, климатических задач, задач физики плазмы, электромагнитных полей, основанных на интегро-интерполяционных, сеточно-характеристических и вариационных методах в многопроцессорных системах используются различные подходы и большой набор инструментов, например: Nast ran; ProCAS; Логос; Wolfarm Mathematics; Anasys 3D; Ansis Maxwell; Fluid Dynamics; Flow vision; Tesseral PRO; Abaqus; Madagaskar; Open Foam; Fierdrake; Matlab; Mathematics; Maple; MathCAD. Для проведения анализа больших данных используется метод графового моделирования, описанный в [31].

Далее приводится описание метода моделирование биологических процессов лечения сетчатки глаза и вычислительной геометрии с использованием современных сервисных и компонентных моделей SOA, SCA WWW3C.

4. Численное моделирование динамических процессов в биологии

В работе [32] описана теория моделирования метода механики сплошной среды применительно в задаче лечения катаракты глаза. Катаракта – это заболевание глаз человека, связанное с помутнением хрусталика. Единственным способом его лечения является удаление хрусталика. Различают интракапсулярное и экстракапсулярное удаление катаракты. В первом случае хрусталик удаляют вместе с капсулой, в которой он находится, а при втором удаляют только хрусталик, а капсула остается в глазу. Среди экстракапсулярных методик наиболее прогрессивной является лазерная экстракция катаракты. Для ее проведения в глаз вводится иголка с расположенным в ней оптоволокном.

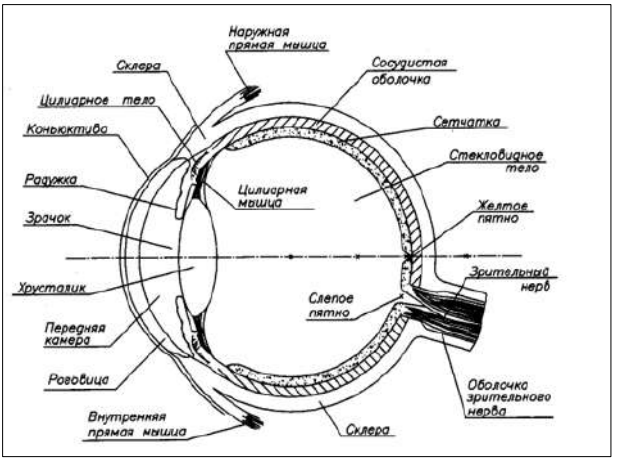

Рис.1. Схема строения глаза человека
Fig. 1. Scheme of the structure of the human eye

При диаметре прокола менее 3 мм можно обойтись без наложения швов. В этом случае могут возникнуть побочные явления и послеоперационных осложнений. Во время лазерного излучения выделяется тепло, воздействующее на роговицу. Кроме того, происходит быстрое расширение хрусталика, имеющее характер микроразрыва. Это приводит к появлению

возмущений в биосреде глаза, распространяющихся к его границам, в частности, к сетчатке, что делает ее зоной риска. Далее кратко описана технология проведения хирургической операции на глазе, схема которого приведена на рис.1.

Глаз имеет не совсем правильную шаровидную форму. Длина его сагиттальной оси в среднем равны 24 мм, горизонтальной – 23,6, вертикальной – 23,3 мм. Глазное яблоко состоит из трех оболочек: наружной, или фибиозной; средней, или сосудистой; внутренней, или сетчатки. Наружная оболочка достаточно тонкая (0,3 – 1 мм), но плотная. Она определяет форму глаза, выполняет защитную функцию места крепления глазодвигательных мышц. Фиброзная оболочка подразделяется на два неравных отдела – роговицу и склеру.

Внутреннее ядро глаза состоит из прозрачных светопреломляющих сред: стекловидного тела, хрусталика и водянистой влаги, наполняющей глазные камеры (передняя – от роговицы до радужки, задняя – от радужки до цилиарного тела). Водянистая влага представляет собой прозрачную жидкость плотностью 1,005–1,007 с показателем преломления 1,33. Количество влаги у человека не превышает 0,2–0,5 мл. Вырабатываемая цилиарным телом водянистая влага содержит соли, аскорбиновую кислоту, микроэлементы. Хрусталик состоит из хрусталиковых волокон, составляющих вещество хрусталика, и сумки-капсулы. Диаметр хрусталика 9–10 мм. Переднезадний размер – 3,5 мм. Стекловидное тело заполняет полость глазного яблока, за исключением передней и задней камер глаза, и таким образом способствует сохранению его формы [28-29].

Объем стекловидного тела взрослого человека 4 мл. Оно состоит из плотного остова и жидкости, причем на долю воды приходится около 99% всего состава стекловидного тела. Физически стекловидное тело представляет собой прозрачный гель, состоящий из внеклеточной жидкости с коллгеном и гиалуровой кислотой в коллоидном растворе.

Частота импульсов лазера – 10Гц, а сам импульс представляет собой цуг длительностью 250 мкс. с модулированной добротностью. Период микроимпульсов в цуге составляет 12,5 мкс, а длительность микроимпульса – 3 мкс. Обсчитывались только первые 250 мкс, поскольку полный расчет одного лазерного импульса потребовал бы сделать $2 \cdot 10^7$ итераций. Кроме того, интенсивность возмущения заметно затухает после прекращения воздействия.

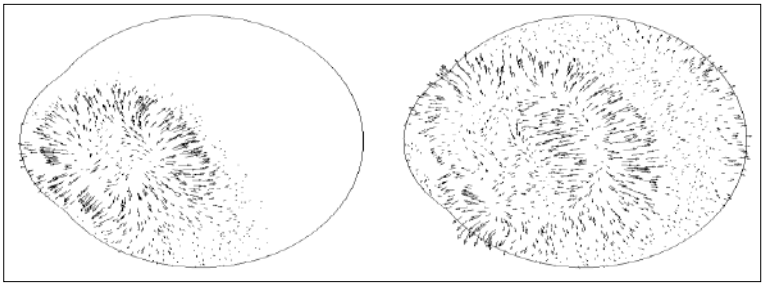

Рис. 2. Поле скоростей, интенсивность и направление скоростей частиц среды глаза в моменты времени: $\boldsymbol{t_1 = 6,24}$ мкс *и* $\boldsymbol{t_2 = 23,6}$ мкс

Fig. 2. Velocity field, intensity and direction of the velocities of the particles of the medium of the eye at times $\boldsymbol{t_1 = 6,24\ \mu s}$ *и* $\boldsymbol{t_2 = 23,6\ \mu s}$

На рис. 2 приведено распределение скоростей частиц среды глаза в моменты времени $t_1 = 6{,}24$ мкс, когда возмущение достигло радужки, и $t_2 = 23{,}6$ мкс, при котором возмущение отразилось от задней поверхности глаза. Можно отметить, что движение среды глаза имеет достаточно сложный характер. В начальный момент времени под воздействием внешнего источника идет расширение биосреды, после прекращения воздействия в зоне хрусталика давление становится меньше, чем в окружающих его областях. Это вызывает изменение направления скоростей на противоположные и как следствие – последующее сжатие глаза.

Под углом 60^0 к оптической оси глаза вводится световод, по которому в хрусталик вводится лазерное излучение. Интенсивность лазерного излучения рассчитывается по закону Ламберта-Бера:

$$Q(r,h) = I(r)e^{-H/h}, I(R) = I_0 e^{-(r/R)^2},$$

где r – радиус облучаемой зоны, h – вертикальная координата, H – глубина поглощения ($H = 3.2$ мм), I_0 – интенсивность в центре световода, R – диаметр световода ($R = 0{,}3$ мм). Полная энергия излучения бралась 250 м.

Для математического описания поведения биосреды глаза использовались уравнения динамики деформируемой среды в следующем виде:

$$\rho\dot{v}_i - \nabla_j\sigma_{ij} = 0; i,j = 1{,}2;\ \dot{\sigma}_{ij} = q_{ijkl}\dot{\varepsilon}_{kk} - \frac{\gamma\delta_{ij}}{\rho c}Q - \frac{S_{ij}}{\tau_0}; \dot{\Theta} = -\frac{\gamma}{\rho c}\dot{\varepsilon}_{kk} + \frac{S_{kl}S_{kl}}{\mu\tau_0} + \frac{Q}{\rho c};$$

здесьQ_0 – плотность объемных источников энергии, Θ – коэффициент теплопроводности,

$$q_{ijkl} = \lambda\sigma_{ij}\sigma_{kl} - \frac{\gamma\delta_{kl}\delta_{ij}}{\rho c} + \mu\left(\delta_{ik}\delta_{jl} + \delta_{jk}\delta_{il}\right).$$

Физические константы для каждого элемента глаза полагались следующими (μ, λ – коэффициенты Лямэ, ρ – плотность, c – удельная теплоемкость, α – коэффициент линейного расширения, τ – время релаксации).

Во время операции глаз периодически расширялся и сжимался. При этом в зонах расширения возможны разрывы тканей и появление кавитационных процессов. Кроме того, на рассматриваемую волновую картину оказывают влияние появление отраженных волн от свободной границы глаза и границ материалов глаза (рис. 2).

На основании данной модели можно оптимизировать режимы работы лазера с тем, чтобы снизить травмирующее воздействие на ткани как переднего, так и заднего сегментов глаза.

В целом возмущения в области хрусталика локализуются и ослабеваются по мере удаления от внешнего источника. Это вполне согласуется с клиническими данными, так как заметных разрушений вне хрусталиковой сумки обычно не наблюдается. Более слабые, возмущения достигаются для сетчатки и зрительного нерва и области роговицы.

В результате моделирования получены численные характеристики происходящих динамических процессов при лазерной экстракции катаракты. На волновую картину оказывают влияние появление отраженных волн от свободной границы глаза и границ материалов глаза (рис. 3).

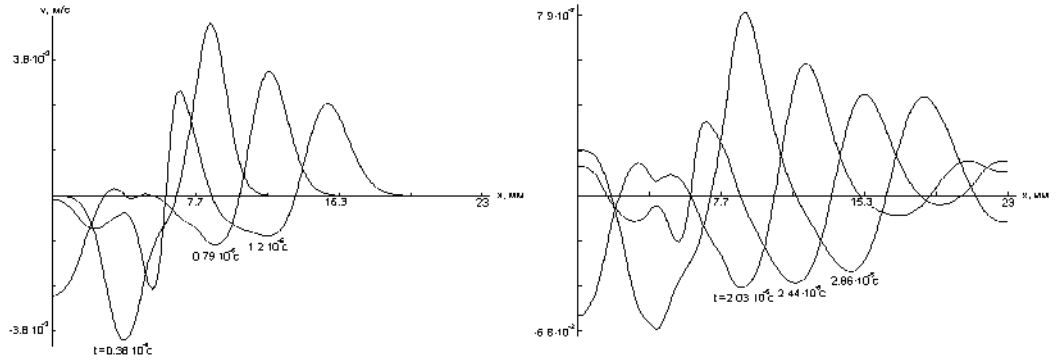

Рис. 3. Проекция значений компонент скоростей на оптическую ось глаза в различные моменты времени

Fig. 3. Projection of the values of the velocity components on the optical axis of the eye at different points in time

Таким образом, моделирование и выполнение операций на хрусталике глаза методом лазерной экстракции катаракты было успешно проведено.

5. Математическое моделирование предмета «Вычислительная геометрия»

5.1 Определение Вычислительной геометрии

Шедевром прикладной математики является античная геометрия Евклида. Главным его вкладом в геометрию, изложенным в «Началах» является: аксиоматический метод доказательства утверждений. Евклидова схема состоит из алгоритма и его доказательства. Евклидова геометрия включает набор допустимых инструментов (линейка, циркуль) и множество допустимых операций, которые можно выполнить с их помощью. Позднее Архимед предложил корректную конструкцию для задачи трех секций угла в 60°, добавив новые примитивы. Абель в 1828 году предложил найти корень любого алгебраического уравнения, используя только арифметические операции. Задача определения количественной меры сложности была сформулирована в 1902 году Лемуаном. Гилберт сформулировал необходимое и достаточное условия вычислимости f с использованием n операций при извлечении квадратного корня. В 1972 году Джордж Мор определил, что любое построение, которое осуществляется с помощью циркуля и линейки, можно выполнить только циркулем в случае, когда искомые объекты определяются точками. Таким образом, была выдвинута идея моделирования с использованием линейки и конечного числа операций циркуля.

Название «*Вычислительная геометрия*» впервые официально появилось на русском языке в связи с выходом монографии [33]. В этой монографии шла речь о задачах геометрического моделирования. Спустя шесть лет на русском языке вышла монография [34] с близким названием, но в ней шла речь об анализе оценок сложности и построении эффективных комбинаторных алгоритмов для решения геометрических задач. Оба направления имеют глубокие геометрические корни и непосредственную связь с ЭВМ. На сегодняшний день за первым направлением закрепилось название «геометрическое моделирование», а за вторым – «вычислительная геометрия».

*Определение***.** Вычислительная геометрия – это наука, предметом исследования которой является анализ и построение эффективных алгоритмов решения геометрических задач, оценки их сложности средствами теории алгоритмов.

Вычислительная геометрия (computational geometry) стала отраслью компьютерных наук для изучения алгоритмов решения задач геометрии на ЭВМ. Основой развития вычислительной геометрии как дисциплины стала компьютерная графика и системы автоматизированного проектирования. Многие задачи вычислительной геометрии являются классическими и могут использоваться при математической визуализации геометрических объектов. Другим важным применением вычислительной геометрии является робототехника (задачи распознавания образов, планирование их движениями), геоинформационные системы (геометрический поиск, планирование маршрута), дизайн микросхем, программирование станков с числовым программным управлением и др.

5.2 Основные методы и задачи вычислительной геометрии

Основными разделами предмета вычислительной геометрии являются следующие.

- *Комбинаторная вычислительная геометрия*, или алгоритмическая геометрия, которая рассматривает геометрические объекты как дискретные сущности. Основополагающим материалом является книга [33], в которой впервые в 1975 был использован термин «вычислительная геометрия» в таком определении.
- *Численная вычислительная геометрия*, или машинная геометрия – это геометрическое моделирование, которое имеет дело в основном с представлением объектов реального мира в форме пригодной для дальнейшей компьютерной обработки. Этот раздел можно рассматривать как дальнейшее развитие вычислительной геометрии и часто рассматривается как раздел компьютерной графики.

Комбинаторная вычислительная геометрия содержит разработанные эффективные алгоритмы и структуры данных для решения задач в терминах базовых геометрических объектов: точки, отрезки, многоугольники, многогранники и др.

Вычислительную геометрию используют для работы над очень большими наборами данных с десятком или сотен миллионов точек. В вычислительной геометрии различают статические задачи и задачи геометрического поиска.

К *статическим задачам* относятся следующие:

- *выпуклая оболочка с* набором точек, необходимых для нахождения наименьшего выпуклого многоугольника, содержащего все точки;
- *пересечение отрезков* в наборе отрезков;
- *триангуляция Делона*;
- *диаграмма Вороного,* содержащая набор точек, разделяющих пространство на сектора, каждая точка которых ближе к набору других;
- *задача ближайшей пары точек* из набора точек для нахождения кратчайшего расстояния;
- *евклидов кратчайший путь,* который соединяет две точки в Евклидовом пространстве (с полигональными препятствиями) кратчайшим образом.
- *триангуляция многоугольника* – разбивка на внутренние треугольники.

В этих задачах входные данные состоят из двух частей: пространство поиска и запросов, которые разнятся в различных видах задач [34]. Для обеспечения эффективного выполнения нескольких запросов пространство поиска проводится предварительная обработка вида:

- *региональный поиск* с целью эффективного поиска и обработки набора точек, содержащихся в запрошенном регионе;
- *локализация точки,* основанная на разбиении пространства на регионы с целью определения заданной точки в нахождения ее определенном регионе;
- *поиск ближайшего соседа* среди набора точек пространства;
- *трассировка лучей* в заданном наборе объектов пространства лучей, которые пересекают запрошенный луч.

Среди этих классов методов рассматривается метод локализации точек в простых и выпуклых многоугольниках.

Динамические задачи – это задачи, входные данные которых меняются (добавляются или удаляются). Алгоритмы решения таких задач основываются на динамических структурах данных. Любую задачу вычислительной геометрии можно решать динамически, но за счет дополнительных вычислительных ресурсов. Т.е. региональный поиск, построение выпуклой оболочки можно проводить над множеством точек, которые изменяются. Вычислительная сложность этого класса задач задается такими параметрами:

- ресурсы, необходимые для определения структуры данных при поиске;
- ресурсы, необходимые для модификации построенной структуры и задания ресурсов для получения ответа на запросы.

К динамическим методам вычислительной геометрии относятся методы регионального поиска, основанные на методах локализации точек на простых и выпуклых многоугольниках:

- метод Грехема и Джарвиса для обхода точек в простых и выпуклых многоугольниках, пересеченных прямыми линиями, для решения задач с использованием арифметических операций и уравнений;
- метод быстрой сортировки на множестве точек и линий пересечения выпуклых и невыпуклых фигур с системой координат;
- метод М.Шеймоса по структурной организации данных для выпуклых многоугольников на множестве точек, а также использования описания данных по методу Препарата, в соответствующих точках внутри и вне выпуклых оболочек. Произошел переход к новым графовым структурам Штейнера для решения разных задач триангуляции и поиска

наближайшего соседа и др.

Вариации. Во многих программах задачи вариации рассматривается как задачи первого класса. Тем не менее, во многих случаях нужно определить курсор мышки внутри данного многоугольника. Курсор постоянно перемещается, а многоугольник не меняется. Аналогично можно проверять определенный летательный аппарат, который изображен на экране радара, и не пересек границу страны. Такие задачи можно считать задачами геометрического запроса. В CAD-системах сам многоугольник может варьироваться, поэтому задача является динамической. Таким образом, в описание онтологической схемы вычислительной геометрии входят: комбинаторная вычислительная геометрия и классы методов: статистические, динамические методы и задачи вариации.

5.3 Моделирование графовой онтологической схемы вычислительной геометрии

Сущеествует целый ряд инструментов поддержки онтологий. Наибольшую популярность приобрел инструмент Protege. Этот инструмент основан на фреймовая модели представления знаний и редактирования моделей, описанных в разных форматах UML, XML, SHOE, DAML, OML, RDF и RDFS, и др. В Protege имеются готовые плагины: Protege OWL Plugin, подобный тестам JUnit; средства генерации описаний в языке Java и возможность преобразовать в формат XML-схемы на платформе Eclipse в XML Schema Infoset Model (XSD) [13-14, 19].

В Protege 4.1 онтология представляется классами, слотами, фасетами и аксиомами. Классы описывают базовые понятия предметной области, а слоты — их свойства. Фасеты описывают свойства слотов (конкретные типы и возможные диапазоны значений). Аксиомы определяют дополнительные ограничения (правила). Классы могут быть абстрактными или конкретными. Абстрактные классы являются контейнерами конкретных классов и могут содержать абстрактные атрибуты. Конкретные классы содержат слоты, которым могут быть значения атрибутов. Фасеты и слоты задаются графовыми XML-схемами. Кардинальность слота определяет возможное количество значений слота, ограничение типа значений слота (например, целое и др. экземпляры класса), предельные значения (мин. и макс.) для числовых слотов и т.п.

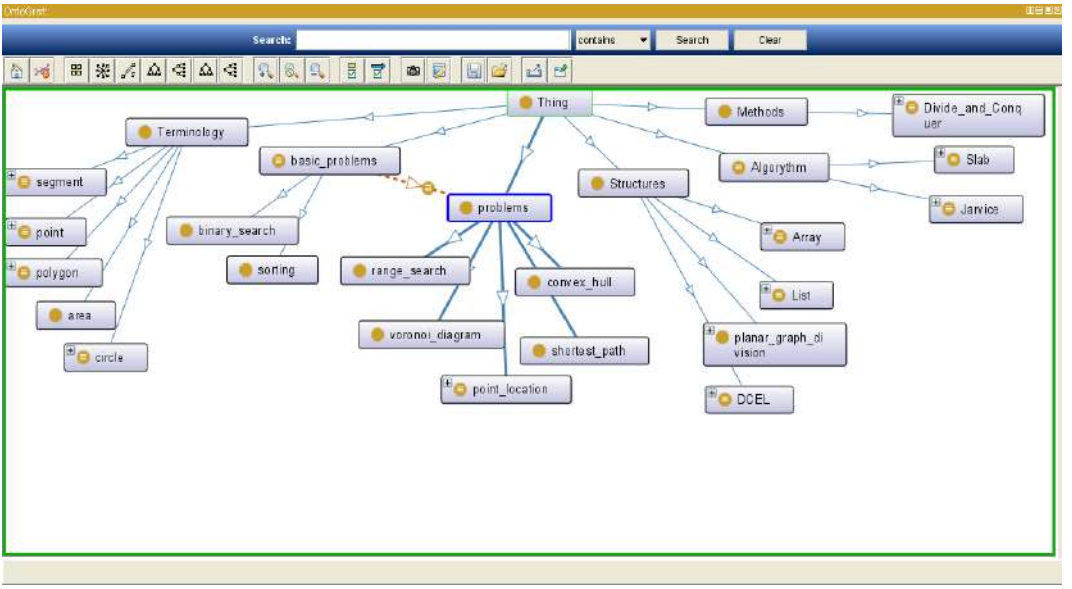

Рис. 3. Онтологическая схема задач вычислительной геометрии
Fig. 3. Ontological scheme of computational geometry problems

На Protege 4.1 было построено описание основных задач «Вычислительная геометрия» в виде онтологической модели (рис. 4). Модель отображает графовую XML-структуру, в вершинах которой находятся названные методы вычисления разных геометрических задач.

Все геометрические задачи, представленные в вершинах графовой схемы онтологии, представлены в языке XML, тексты которых занимают более 15 стр. В МФТИ на кафедре Информатика и вычислительная физика создается сайт «Вычислительная геометрия» [35] для обучения студентов этому предмету, а также теории и технологии программирования [36].

6.Заключение

Описаны подходы к моделированию технических средств задач прикладной математики, проведения их к постановке на ЭВМ путем декомпозиции задач на отдельные функциональные подзадачи, их алгоритмизации и описания средствами математической логики, граф-схемного языка А.А. Ляпунова, адресного языка и языков программирования.

Дана характеристика численного моделирования и программирования задач прикладной математики: баллистики, физического эксперимента, механики сплошных сред, возглавляемыми учеными академиками нашей страны (А.А.Самарским, М.В. Келдышем, О.М. Белоцерковским, А.А. Дородницыным и др.). Рассмотрены программные средства моделирования математических задач прикладных областей (физического эксперимента, ядерного и термоядерного проекта и др.) и дисциплин: алгебра, геометрия, интеллект, медицина, биология, нейро- и нано-технологий и др. Описаны новые многопроцессорные компьютерные системы для решения сложных задач. Дано описание технологии экстракции катаракты глаза и учебных дисциплин «Вычислительная геометрия» и теории технологии программирования в МФТИ, представленных на сайте http://7dragons.ru/ru.

Список литературы / References

[1] В.В. Липаев. Фрагменты истории развития отечественного программирования для специализированных ЭВМ в 50-80-е годы. М., Синтег, 2003, 126 стр. / V.V. Lipaev. Fragments of the history of the development of domestic programming for specialized computers in the 50-80s. M., Sinteg, 2003, 126 p. (in Russian).

[2] В.М. Глушков. Об одном методе автоматизации программирования. Проблемы кибернетики, № 2, 1959, стр. 181-184 / V.M. Glushkov. On one method of programming automation. Problems of Cybernetics, No. 2, 1959, pp. 181-184 (in Russian).

[3] Е.А. Жоголев, Г.С. Росляков, Н.П. Трифонов, М.Р. Шура-Бура. Система стандартных подпрограмм. ГИФМЛ, 1958, 231 стр. / E.A. Zhogolev, G.S. Roslyakov, N.P. Trifonov, M.R. Shura-Bura. System of standard subroutines. GIFML, 1958, 231 p. (in Russian).

[4] А.П. Ершов. Введение в теоретическое программирование. М., Физтехлит, 1977, 286 стр. / A.P. Ershov. Introduction to theoretical programming. Moscow, Fiztekhlit, 1977, 286 p. (in Russian).

[5] С.С. Лавров. Лекции по теории программирования. СПб, Нестор, 1999, 107 стр. / S.S. Lavrov. Lectures on programming theory. SPb, Nestor, 1999, 107 p. (in Russian).

[6] А.А. Самарский, А.П. Михайлов. Компьютеры и жизнь (математическое моделирование). М., Педагогика, 1987, 128 стр. / A.A. Samarsky, A.P. Mikhailov. Computers and Life (Mathematical Modeling). M., Pedagogy, 1987, 128 p. (in Russian).

[7] А.А. Самарский, Ю.П. Попов. Разностные методы решения задач газовой динамики. М., Наука, Физматлит, 1992, 423 стр. / A.A. Samarsky, Yu.P. Popov. Difference methods for solving problems of gas dynamics. Moscow, Nauka, Fizmatlit, 1992, 423 p. (in Russian).

[8] А.А. Самарский. Теория разностных схем. М., Наука, 1977. 656 стр. / A.A. Samarskiy. The theory of difference schemes. Moscow, Nauka, 1977.656 p. (in Russian).

[9] О.М. Белоцерковский. Численное моделирование в механике сплошных сред. М., Наука, Физматлит, 1994, 441стр. / O.M. Belotserkovsky. Numerical modeling in continuum mechanics. M., Nauka, Fizmatlit, 1994, 441p. (in Russian).

[10] О.М. Белоцерковский, А.С. Холодов. Медицина в зеркале информатики. М., Наука, 2008, 242 стр. / O.M. Belotserkovsky, A.S. Kholodov. Medicine in the Mirror of Informatics. M., Nauka, 2008, 242 p. (in Russian).

[11] В.А. Евстигнеев. Применение теории графов в программирование. М., Наука, Редакция физ.-мат. наук, 1985 г., 351 стр. / V.A. Evstigneev. Application of graph theory to programming. M., Nauka, 1985, 351 p. (in Russian).

[12] В.Н. Касьянов, В.А. Евстигнеев. Графы в программировании: обработка, визуализация и применение. СПб., БХВ-Петербург, 2003 г., 1104 стр. / V.N. Kasyanov, V.A. Evstigneev. Graphs in programming: processing, visualization and application. SPb., BHV-Petersburg, 2003, 1104 p. (in Russian).

[13] Е.М. Лаврищева, В.Н. Грищенко. Связь разноязыковых модулей в ОС ЕС. М., Финансы и статистика, 1982 г., 137 стр. / E.M. Lavrischeva, V.N. Grishchenko. Linking multilingual modules in the EU OS. M., Finance and Statistics, 1982, 137 p. (in Russian).

[14] E.M. Lavrischeva. The Theory Graph Modeling and Programming Paradigm of Systems FROM Modules TO the Application Areas. Transactions on Machine Learning and Artificial Intelligence, vol. 7, no 4, 2019, pp. 21-43.

[15] Лаврищева Е.М. Теория объектно-компонентного моделирования программных систем. Препринт ИСП РАН, № 29, 2016 г., 52 стр. / Lavrischeva E.M. The theory of object-component modeling of software systems. ISP RAS Preprint, No. 29, 2016, 52 p. (in Russian).

[16] И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. Теория соответствия для систем с блокировками и разрушениями. Москва, Наука, 2008 г., 412 стр. / B. Burdonov, A.S. Kosachev, V.V. Kulyamin. Correspondence theory for systems with lockings and destructions. Moscow, Nauka, 2008, 412 p. (in Russian).

[17] З.В. Апанович. Визуализация больших графов и матрицы смежности. Труды XX Всероссийской научной конференции «Научный сервис в сети Интернет», 2018 г., стр. 28-41 / Z.V. Apanovich. Visualization of large graphs and adjacency matrices. In Proc. of the XX All-Russian Scientific Conference on Scientific Service on the Internet, 2018, pp. 38-41 (in Russian).

[18] Е.М. Лаврищева, А.К.Петренко. Моделирование семейств программных систем. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 49-64 / Lavrischeva K.M., Petrenko A.K. Software Product Lines Modeling. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 49-64 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-4.

[19] Е.М. Лаврищева. Программная инженерия и технология программирования сложных систем. М., Юрайт, 2019 г., 432стр. / E.M. Lavrischeva. Software engineering and technology of complex systems programming. M., Urait, 2019, 432 p. (in Russian).

[20] Е.Ю. Титаренко. Теория графов. Институт кибернетики Томского политехнического университета, 2016 г., 64 стр. / E.Yu. Titarenko. Graph theory. Institute of Cybernetics, Tomsk Polytechnic University, 2016, 64 p. (in Russian).

[21] E.M. Lavrishcheva, I.B. Petrov. Ways of Development of Computer Technologies to Perspective Nano. Future Technologies Conference (FTC), 2017, pp. 539-548.

[22] Г.И. Марчук. Методы расщепления, М., Наука, 1988 г., 263 стр. / G.I. Marchuk. Splitting methods, M., Nauka, 1988, 263 p.

[23] М.В. Якобовский. Введение в параллельные методы решения задач. М., МГУ, 2012 г., 328 стр. / M.V. Jacobovsky. Introduction to the Methods of Parallel Problem Solving. M., Moscow State University, 2012, 328 pp. (in Russian).

[24] А.Г. Куликовский, Н.В. Погорелов, А.Ю Семенов. Математические вопросы численного решения гиперболических систем уравнений. М., Физматлит, 2012 г., 656 стр. / A.G. Kulikovsky, N.V. Pogorelov, A.Yu Semenov. Mathematical problems in the numerical solution of hyperbolic systems of equations. M., Fizmatlit, 2012, 656 p. (in Russian).

[25] Б. Олдер, С. Фернбах, М. Ротенберг (редакторы). Вычислительные методы в физике плазмы. М., Мир, 1974. 514 стр. / B. Alder, S. Fernbach, M. Rotenberg (editors). Methods in computational physics, vol. 9. Orlando Academic Pr., 1970, 498 p.

[26] Н.Г. Яковлев. Математическое моделирование земной системы. М., МАКС-Пресс, 2016 г., 328 стр. / N.G. Yakovlev. Mathematical modeling of the earth system. M., MAKS-Press, 2016, 328 p. (in Russian).

[27] И.Е. Квасов, В.Б. Левянт, И.Б. Петров. Решение прямых задач сейморазведки в трещиноватых средах методом сеточно-характеристического моделирования. М., Геомодель, 2016 г., 295 стр. / I.E. Kvasov, V.B. Levyant, I.B. Petrov. Solving direct problems of seismic prospecting in fractured media by the method of grid-characteristic modeling. M., Geomodel, 2016, 295 p. (in Russian).

[28] И.Б. Петров. Воздействие льда и воды на оффшорные структуры и прибрежные зоны в Арктике. Сборник трудов Научной сессии общего собрания членов РАН, 16 декабря 2014 г., М., Наука, 2015 г., стр. 230-237 / I.B. Petrov. Impact of ice and water on offshore structures and coastal zones in the Arctic.

In Proc.of the Scientific Session of the General Meeting of Members of the Russian Academy of Sciences, December 16, 2014, M., Nauka, 2015, pp. 230-237 (in Russian).

[29] Дж. Мюррей. Математическая биология. Т. 1. Введение. М., Ижевск, Институт компьютерных исследований, 2009 г., 776 стр. / J.D. Murray. Mathematical Biology: I. An Introduction. Springer, 2007, 574 p.

[30] Г.И. Марчук. Математические модели в иммунологии. М., Наука, 1985 г., 239 стр. / G.I. Marchuk. Mathematical models in immunology. M., Nauka, 1985, 239 p. (in Russian).

[31] Е.М. Лаврищева, А.Г. Рыжов. Применение теории общих типов данных стандарта ISO/IEC 11404 GDT к Big Data. Евразийский Союз Ученых (ЕСУ), no. 31, 2016 г., стр. 99-108 / E.M. Lavrischeva, A.G. Ryzhov. Application of the theory of common data types of the ISO / IEC 11404 GDT standard to Big Data. Eurasian Union of Scientists, no. 31, 2016, pp. 99-108. (in Russian).

[32] И.Б. Петров. Математическое моделирование в медицине и биологии на основе моделей механики сплошных сред. Труды МФТИ, 2009 г., том 1, no 1, стр. 5-16 / I.B. Petrov. Mathematical modeling in medicine and biology based on models of continuum mechanics. Proceedings of MIPT, 2009, volume 1, no 1, pp. 5-16. (in Russian).

[33] А. Фокс, М. Пратт. Вычислительная геометрия. Применение в проектировании и производстве. М., Мир, 1982 г., 304 стр. / I.D. Faux, M.J. Pratt. Computational Geometry for Design and Manufacture. Ellis Horwood, 1980, 329 p.

[34] Ф. Препарата, М. Шеймос. Вычислительная геометрия: Введение. М., Мир, 1989 г., 478 стр. / F.P. Preparata, M.I. Shamos. Computational Geometry: An Introduction. Springer, 1985, 390 p.

[35] Е.М. Лаврищева, Л.Е.Карпов, А.Н. Томилин. Семантические ресурсы для разработки онтологии научной и инженерной предметных областей Труды XVII Всероссийской научной конференции «Научный сервис в сети Интернет», 2016 г., стр. 223-239 / E.M. Lavrischeva, L.E. Karpov, A.N. Tomilin. Semantic resources for the development of ontology of scientific and engineering subject areas. In Proc. of the XVII All-Russian Scientific Conference on Scientific Service on the Internet, 2016, pp. 223-239. (in Russian).

[36] Е.М. Лаврищева. Программная инженерия. Тема 1. Теория программирования. Учебно-методическое пособие. М., МФТИ, 2016 г., 48 стр. / E.M. Lavrischeva. Software engineering. Topic 1. Programming theory. Study guide. M., MIPT, 2016, 48 p. (in Russian).

[37] Е.М. Лаврищева. Программная инженерия. Тема 2. Технология программирования. Учебно-методическое пособие. М., МФТИ, 2016 г., 52 стр. / E.M. Lavrischeva. Software engineering. Topic 2. Programming technology. Study guide. M., MIPT, 2016, 52 p. (in Russian).

Информация об авторах / Information about authors

Екатерина Михайловна ЛАВРИЩЕВА – доктор физико-математических наук, профессор, главный научный сотрудник ИСП РАН, профессор МФТИ, лауреат премии кабинета Министров СССР (1985). Область интересов: надежность и качество, моделирование сложных систем, Web-системы, сборочное программирование.

Ekaterina Mikhailovna LAVRISCHEVA – Doctor of Physical and Mathematical Sciences, Professor, Principal Researcher at ISP RAS, Professor at Moscow Institute of Physics and Technology, Laureate of the USSR Cabinet of Ministers Prize (1985). Areas of interest: reliability and quality, modeling of complex systems, Web-systems, assembly programming.

Игорь Борисович ПЕТРОВ является доктором физико-математических наук, профессором, членом-корреспондентом РАН, заведующим кафедрой информатики МФТИ. В число научных интересов входят сеточно-характеристический метод, математическое моделирование, высокопроизводительные вычислительные системы.

Igor Borisovich PETROV – Doctor of Physical and Mathematical sciences, Professor, Corresponding Member of the Russian Academy of Sciences, Head of the Department of Computer Science at the Moscow Institute of Physics and Technology. His research interests include the grid-characteristic method, mathematical modeling, and high-performance computing systems.

DOI: 10.15514/ISPRAS–2020–32(6)–14

Архитектура программного средства с открытым исходным кодом для численного моделирования потоков на горных склонах

Д.И. Романова, ORCID: 0000-0002-5771-4114 <romanovadi@gmail.com>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1

Аннотация. В настоящей работе разрабатывается архитектура решателя, реализующего новую трехмерную математическую модель для описания динамики потоков на склонах с учётом захвата и отложения материала. Также проводится сравнение двух подходов для описания динамики потоков на склонах: с использованием осреднённых по глубине уравнений механики сплошной среды (уравнений типа мелкой воды) и с использованием трёхмерного моделирования, основанного на полных, не осреднённых по глубине, уравнениях механики сплошной среды. С применением этих двух подходов проведено моделирование экспериментов по спуску потока в лотке и взаимодействию потока с комплексом заградительных сооружений. Проведено сравнение численных решений с экспериментальными данными. Кроме того, оба подхода применены к расчёту снежной лавины в 22-ом лавинном очаге горы Юкспор (Хибины). Дальность выброса лавины и форма лавинных отложений сравнивалась с натурными данными, полученными по результатам измерения реальной лавины, сошедшей в данном очаге. В процессе численного эксперимента были получены распределения таких величин, как скорость потока, глубина, плотность, молекулярная и турбулентная вязкость, значения плотности турбулентной кинетической энергии, диссипации турбулентной кинетической энергии, значение напряжения сдвига на дне потока. С использованием полученных данных разрабатывается математическая модель для описания захвата потоком материала подстилающей поверхности при её разрушении и отложения материала потока на склон. Для реализации полученной математической модели разработана архитектура решателя multiphaseEulerChangeFoam, реализующего трёхфазную многоскоростную модель с фазовыми переходами между материалом подстилающей поверхности и материалом движущегося потока. В качестве основы для разрабатываемого решателя взят классический решатель multiphaseEulerFoam из пакета OpenFOAM.

Ключевые слова: математическое моделирование; численное моделирование; снежная лавина; грязекаменный сель; склоновые потоки; OpenFOAM; faSavageHutterFoam; interFoam; multiphaseEulerChangeFoam; неньютоновская среда; среда Хершеля-Балкли; многофазный поток; турбулентный поток; k-ε модель турбулентности; уравнения мелкой воды

Для цитирования: Романова Д.И. Архитектура программного средства с открытым исходным кодом для численного моделирования потоков на горных склонах. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 183-200. DOI: 10.15514/ISPRAS–2020–32(6)–14

Благодарности: Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-31-90105

Architecture of Open Source Program for Numerical Modeling of Flows on Mountain Slopes

D.I. Romanova, ORCID: 0000-0002-5771-4114 <romanovadi@gmail.com>

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

Abstract. In this paper, we compare two approaches to describe the dynamics of flows on mountain slopes using the depth-averaged equations of continuum mechanics and using the complete, not depth-averaged, equations of continuum mechanics for three-dimensional modeling. Using these two approaches, a simulation of an experimental slush flow in the tank and the interaction of the flow with dam barrier protection was carried out. Numerical solutions are compared with experimental data. Also, both approaches are applied to the calculation of an avalanche in the 22nd avalanche cite of Mount Yukspor (Khibiny). Avalanche run-out distance and the shape of the avalanche deposits are compared with field data obtained from the measurement of a real avalanche. In the course of a numerical experiment, distributions of such quantities as flow velocity, depth, density, molecular and turbulent viscosity, values of the density of turbulent kinetic energy, dissipation of turbulent kinetic energy, and shear stress at the bottom of the flow were obtained. Using the obtained data a mathematical model is developed to describe the entrainment of the underlying material by the flow during slope erosion and the deposition of the flow material on the slope. To implement the obtained mathematical model, the architecture of the multiphaseEulerChangeFoam solver was developed, which implements a three-phase multi-velocity model with phase exchange between the material of the underlying surface and the material of the flow. The classic solver multiphaseEulerFoam from the OpenFOAM package is taken as a basis for the developed solver.

Keywords: mathematical modeling; numerical modeling; snow avalanche; mudflow; slope flow; OpenFOAM; faSavageHutterFoam; interFoam; multiphaseEulerChangeFoam; non-Newtonian fluid; Herschel-Bulkley fluid; multiphase flow; turbulent flow; k-ε turbulence model; shallow water equations

For citation: Romanova D.I. Design of open source software architecture for numerical modeling of flows on mountain slopes. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 183-200 (in Russian). DOI: 10.15514/ISPRAS-2020-32(6)-14

Acknowledgements. The reported study was funded by RFBR, project number 19-31-90105.

1. Введение

Потоки на склонах гор, такие как лавины, грязекаменные сели, оползни, представляют большую опасность для жизнедеятельности людей, разрушают объекты инфраструктуры, сельского хозяйства, создают помехи для автомобильного и железнодорожного сообщения. Эти явления изучаются уже больше века и было разработано много методик для прогнозирования лавин и селей, оценки опасности территорий, построения заградительных, отводящих и удерживающих сооружений, предупредительных спусков лавин меньшей мощности. Однако, эти смертоносные явления в горах продолжают происходить и приводят к большому материальному ущербу и гибели десятков людей. Начиная с 2010 года, можно перечислить следующие катастрофы:

- 1 января 2010 года оползни, вызванные проливными дождями на острове Илья Гранди бразильского штата Рио-де-Жанейро, привели к гибели более 50 человек;
- 24 октября 2011 года сильные грозы в северной области Лигурия и центральной Тоскане спровоцировали мощные селевые потоки и оползни, что привело к гибели десяти человек;
- 14 октября 2012 года селевой поток разрушил школу в провинции Юньнань в Китае, погибли 18 школьников;

- Барсемская селевая катастрофа в июле 2015 года в Таджикистане [1] привела к очень большому материальному ущербу;
- в феврале 2016 года лавина с горы Юкспор, рассмотренная в данной работе унесла жизни 3 людей, завалила автомобильную и производственную железную дороги, выбила окна в близлежащем жилом доме;
- 1 сентября 2017 года пять сёл в Эльбрусском районе были отрезаны селем от мира, когда из-за сильных дождей произошел прорыв высокогорных ледниковых Башкаринских озёр [2];
- 19 января 2017 года отель Rigopiano в Италии накрыт лавиной, сошедшей с горы Гран-Сассо в регионе Абруццо.

Этот список можно ещё долго продолжать, однако приведённых примеров достаточно, чтобы утверждать, что проблемы лавинной и селевой безопасности требуют более глубокого и детального исследования.

Математическое моделирование динамики потоков на склонах гор позволяет спрогнозировать опасные зоны, изучить эрозию склона или русла потоком, рассчитать нагрузку на заградительные, удерживающие или отводящие сооружения.

В данной работе исследуется перспективность применения двух различных подходов для моделирования снежных лавин, грязекаменных селей, оползней: с помощью осреднённых по глубине уравнений механики сплошной среды и полного трёхмерного подхода с использованием уравнений механики сплошной среды для многофазной среды. На самом деле подходов для моделирования потоков на склонах намного больше, их обзор представлен, например, в работах [3], [4]. Для сравнения взяты самый популярный в настоящее время подход (с использованием осреднённых по глубине уравнений) и самый информативный (с использованием уравнений механики сплошной среды для трёхмерного моделирования динамики потока).

1.1 Обзор использования осреднённых по глубине уравнений механики сплошной среды для описания динамики потоков на склонах

Осреднённые по глубине уравнения механики сплошной среды используются для описания потоков, в которых линейный масштаб вдоль склона L много больше их характерной глубины h (приближение мелкой воды). Для многих потоков на горных склонах это условие выполняется. Эти уравнения выводятся интегрированием по глубине вдоль нормали к склону (или – для потоков в руслах и лотках – по поперечному сечению) полных уравнений механики сплошных сред. При выводе пренебрегается некоторыми членами, которые считаются малыми в силу малости отношения h/L , и делаются дополнительные предположения о виде членов, описывающих трение и массообмен на дне и на верхней границе потока. Таким образом уменьшается размерность задачи и, в классическом варианте, решается уже не трёхмерная, а двумерная задача в проекции на склон.

Данный подход в основном используется в настоящее время наряду с моделями материальной точки. Уравнения мелкой воды реализованы во многих коммерческих программных кодах, используемых для моделирования склоновых потоков, например, пакетах DAN [5], SamosAT [6], [7], FLATModel [8], RAMMS [9].

Есть также открытые альтернативы данным программным продуктам, например, пакеты TITAN2D [10], [11 и r.avaflow [12], [13], моделирование потоков на склоне с использованием пакета GERRIS [14], решатель faSavageHutterFoam пакета OpenFOAM [15], [16], который используется в данной работе.

Стоит отметить, что данный метод моделирования с использованием уравнений мелкой воды подходит в основном для плотных потоков, то есть плотных лавин, грязекаменных селей и оползней. Также он не позволяет рассчитывать прочность заградительных сооружений, так

как не может дать информации о трёхмерном распределении нагрузки на исследуемый объект, а лишь некоторые интегральные характеристики [17].

1.2 Обзор использования уравнений полных, не осреднённых по глубине, уравнений механики сплошной среды для трёхмерного моделирования динамики потоков на склонах

Второй подход – трёхмерное моделирование динамики потоков на склонах гор – не получил широкого распространения из-за очевидной вычислительной сложности такой задачи, так как в длину поток может достигать нескольких километров, а в толщину порядка нескольких десятков метров. Помимо этого, при трехмерном подходе есть проблема формулировки реологических соотношений для материалов потоков, проблемы моделирования турбулентности для потоков сред со сложной реологией. Однако есть ряд задач, в которых невозможно обойтись моделью мелкой воды, например, расчёт заградительных, удерживающих и отводящих сооружений. Для расчета сил, действующих на конструкции при ударе потока, необходимо знать распределение скорости и давления, в том числе, по глубине потока. Модель мелкой воды не может дать таких распределений, так как в ней рассматриваются лишь осреднённые по глубине параметры потока.

Второй важной задачей, для решения которой нужен трёхмерный подход, является вывод физически обоснованных формул для членов уравнений, описывающих трение на дне, а также скорость вовлечения потоком подстилающего материала и отложение материала потока на склон. Эмпирические формулы, используемые в настоящее время, содержат коэффициенты, не связанные непосредственно со свойствами движущегося материала и характеристиками склона. Они определяются для каждого региона только путем обратных расчетов потоков, для которых имеются данные измерения их параметров.

Существует несколько реализаций трёхмерного моделирования динамики потоков на склонах; например, в работе Ямагучи (Yuya Yamaguchi) и соавторов [18] используется метод конечных элементов на основе подхода Петрова-Галёркина. В работе склоновый поток (снежная лавина) рассматривается как двухфазный (снег и воздух), снег представлен бингамовской жидкостью.

В работе Оды (Kenichi Oda) и соавторов [19] также используется двухфазный подход для описания динамики снежной лавины, только в отличие от работы [18] система уравнений решается конечно-разностным методом.

В работе [20] двухфазный подход для моделирования лавины решается с помощью метода конечных объёмов с использованием пакета OpenFOAM. Именно данный метод будет исследован в данной работе.

Стоит отметить, что при использовании многофазного подхода для трёхмерного моделирования динамики потока на склоне сильно расширяется область применимости по сравнению с предыдущим подходом (с использованием уравнений мелкой воды). Таким образом, можно исследовать не только лавины из мокрого снега, но и снежно-пылевые лавины, всевозможные селевые и оползневые потоки, а также мутьевые потоки под водой.

1.3 Использование комбинированных методов

В силу вышеперечисленных обзоров использования двух, рассмотренных в настоящей статье, подходов, наиболее эффективным предполагается комбинировать оба подхода таким образом, чтобы расчёт динамики потока большого масштаба проводился с помощью осреднённых по глубине уравнений механики сплошной среды, а выделенные, особо важные области, например, взаимодействие с препятствиями, моделировались с помощью полных, не осреднённых по глубине, уравнений механики сплошной среды.

2. Математическая модель потока на склоне с использованием уравнений мелкой воды

Для описания динамики потоков снега, воды, камней, грязи на склонах в данной работе используются модифицированные уравнения мелкой воды, выведенные Раутером (Matthias Rauter) и Туковиком (Zeljko Tukovic) [15]. Эти уравнения схожи с предложенными для расчёта снежных лавин Григоряном, Эглит и Якимовым [3], [21], и Севеджем (Stuart Savage) и Хуттером (Kolumban Hutter) [22], [23] для гранулированных потоков. В работах [21]–[23] и многих других используется криволинейная система координат вдоль склона, и параметры потока рассчитываются в этой, связанной со склоном системе координат. Математическая модель Раутера и Туковика, используемая в данной работе отличается от [21]–[24] тем, что она записывается в глобальной декартовой системе координат, не привязанной к склону. Такой подход позволяет с меньшими вычислительными затратами учитывать влияние кривизны склона на динамику потока.

Выпишем систему уравнений, описывающих движение потока:

$$\begin{cases} \dfrac{\partial h}{\partial t} + \boldsymbol{\nabla} \cdot (h\overline{\mathbf{u}}) = 0, \\ \dfrac{\partial (h\overline{\mathbf{u}})}{\partial t} + \boldsymbol{\nabla}_t \cdot (h\overline{\mathbf{u}\mathbf{u}}) = -\dfrac{1}{\rho}\boldsymbol{\tau}_b + h\mathbf{g}_t - \dfrac{1}{2\rho}\boldsymbol{\nabla}_t(hp_b), \\ \boldsymbol{\nabla}_n(h\overline{\mathbf{u}\mathbf{u}}) = h\mathbf{g}_n - \dfrac{1}{2\rho}\boldsymbol{\nabla}_n(hp_b) - \dfrac{1}{\rho}\mathbf{n}_b p_b. \end{cases} \quad (1)$$

Здесь горизонтальной чертой над символом обозначается осреднение по глубине потока по нормали к склону, h – глубина потока, $\overline{\mathbf{u}}$ – средняя по глубине потока скорость, ρ – плотность потока, $\boldsymbol{\tau}_b$ и p_b – касательное напряжение и давление на дне, $\mathbf{g}$ – ускорение свободного падения, индексом t обозначается проекция на касательную к склону плоскость, индексом n – на нормаль. Первое уравнение системы (1) представляет собой осреднённое по глубине уравнение неразрывности, второе и третье уравнения системы (1), представляют собой нормальную и касательную к склону составляющие осреднённого по глубине закона сохранения количества движения, записанные в векторном виде (такой подход имеет преимущества при реализации методом конечного объема):

$$\frac{\partial (h\overline{\mathbf{u}})}{\partial t} + \boldsymbol{\nabla} \cdot (h\overline{\mathbf{u}\mathbf{u}}) = -\frac{1}{\rho}\boldsymbol{\tau}_b + h\mathbf{g} - \frac{1}{2\rho}\boldsymbol{\nabla}(hp_b) - \frac{1}{\rho}\mathbf{n}_b p_b. \quad (2)$$

Последнее уравнение системы (1) получается посредством умножения уравнения (2) на $\mathbf{n}_b$, тогда первый член зануляется, так как коллинеарен склону; по аналогичному принципу равен нулю член с трением на дне, так как он также действует в плоскости склона. Далее полученное выражение дополнительно домножается на вектор нормали к поверхности склона $\mathbf{n}_b$ для получения уравнения на давление с учётом его направления $\mathbf{n}_b p_b$. Для получения закона сохранения количества движения в тангенциальном направлении уравнение на давление вычитается из (2).

Подробный вывод уравнений системы (1) представлен в работе [15].

В уравнениях (1) вводится градиент по направлению в векторном виде:

$$\boldsymbol{\nabla}_t = (\mathbf{I} - \mathbf{n}_b\mathbf{n}_b) \cdot \boldsymbol{\nabla}, \quad \boldsymbol{\nabla}_n = (\mathbf{n}_b\mathbf{n}_b) \cdot \boldsymbol{\nabla}. \quad (3)$$

Проекции вектора гравитационного ускорения $\mathbf{g}_t$ и $\mathbf{g}_n$ вычисляются аналогичным [grad] методом (вместо оператора набла $\boldsymbol{\nabla}$ подставляется вектор гравитационного ускорения $\mathbf{g}$).

Произведение векторов надо понимать, как внешнее (тензорное) произведение:

$$\mathbf{u}\mathbf{v} = \mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T$$

В модели используется замыкающее соотношение для трения на дне $\boldsymbol{\tau}_b(\mathbf{x}_b)$. При использовании осреднённых по глубине уравнений $\boldsymbol{\tau}_b$ обычно считается функцией h, $|\overline{\mathbf{u}}|$, которые зависят от $\mathbf{x}_b$ и времени. Этот член является функцией координаты точки на склоне $\mathbf{x}_b$. Существует много различных моделей, описывающих трение на дне, среди которых модель Фелми (Adolf Voellmy) [25], модель Кристена (Marc Christen) и соавторов [9], использованная в работе [26], модель Сэвиджа-Хуттера [22], [23]. В данном исследовании используется модель, аналогичная модели Фелми [25]:

$$\boldsymbol{\tau}_b = \mu p_b \frac{\overline{\mathbf{u}}}{|\overline{\mathbf{u}}| + u_0} + \frac{\rho g}{\xi} |\overline{\mathbf{u}}| \overline{\mathbf{u}}. \quad (4)$$

Здесь μ и ξ – константы, зависящие от размера, типа и других параметров потока на склоне. Параметр u_0 – малая величина, введённая, чтобы исключить деление на нуль при осуществлении расчетов.

3. Математическая модель динамики потока на склоне с использованием многофазного подхода

При этом подходе поток рассматривается как многофазное течение, осреднение по глубине не используется. Одна из фаз – воздух, другая – материал потока (снег или грязекаменная смесь), третьей фазой может быть материал подстилающей поверхности. В данной работе рассматривается двухфазная модель – поток-воздух. Используется $k-\varepsilon$ модель турбулентности, основанная на работах [27], [28].

В работе используется метод VOF (объём жидкости) для отслеживания границы свободной поверхности, который был предложен Хиртом и Николсом в 1981 году [29]. Данный метод не отслеживает границу явно, она задаётся как пороговое значение для объёмной доли фазы.

Выпишем систему уравнений для описания двухфазной модели течения, в которой каждая из фаз считается несжимаемой и обе фазы имеют единую скорость (5). Данная модель состоит из следующих уравнений: уравнение неразрывности, уравнения переноса объёмной доли одной из фаз, уравнения сохранения количества движения, уравнение для расчёта турбулентной кинетической энергии и уравнение диссипации турбулентной кинетической энергии.

$$\begin{cases} \nabla \cdot \overline{\mathbf{u}} = 0, \\ \dfrac{\partial \alpha}{\partial t} + \nabla \cdot (\overline{\mathbf{u}} \alpha) = 0, \\ \dfrac{\partial (\rho \overline{\mathbf{u}})}{\partial t} + \nabla \cdot (\rho \overline{\mathbf{u}\mathbf{u}}) = -\nabla \overline{p} + \nabla \cdot \overline{\boldsymbol{\tau}} + \rho \overline{\mathbf{f}}, \\ \dfrac{\partial (\rho k)}{\partial t} + \nabla \cdot (\rho \overline{\mathbf{u}} k) = \nabla \cdot (\mu \nabla k) - \nabla \cdot \left(\dfrac{\mu_t}{\sigma_k} \nabla k + P_k \right) - \rho \varepsilon, \\ \dfrac{\partial (\rho \varepsilon)}{\partial t} + \nabla \cdot (\rho \overline{\mathbf{u}} \varepsilon) = C_{\varepsilon 1} P_k \dfrac{\varepsilon}{k} - \rho C_{\varepsilon 2} \dfrac{\varepsilon^2}{k} + \nabla \cdot \left(\dfrac{\mu_t}{\sigma_\varepsilon} \nabla \varepsilon \right). \end{cases} \quad (5)$$

Здесь $\mathbf{u}$ – скорость смеси; α – объёмная доля выбранной фазы; $\overline{\boldsymbol{\tau}} = 2\mu_{eff}\overline{\mathbf{s}}$ – тензор напряжений, выраженный через тензор скоростей деформации $\overline{\mathbf{s}} = 0.5[\nabla \overline{\mathbf{u}} + (\nabla \overline{\mathbf{u}})^T]$, горизонтальной чертой над буквами обозначается осреднение по Рейнольдсу; $\mu_{eff} = \mu + \mu_t$ – эффективный коэффициент вязкости, сумма молекулярной вязкости и турбулентной, последняя вычисляется по формуле $\mu_t = \rho C_\mu k^2/\varepsilon$; ρ – плотность смеси; $\overline{p}$ – давление; $\overline{\mathbf{f}}$ – плотность массовых сил; k – плотность турбулентной кинетической энергии; ε – диссипация плотности турбулентной кинетической энергии; P_k – скорость производства турбулентной

кинетической энергии средним течением, $C_{\varepsilon 1}$, $C_{\varepsilon 2}$, C_{μ}, σ_k, σ_{ε} – коэффициенты модели турбулентности.

Объёмная доля фазы принимает значения в диапазоне $0 \leq \alpha \leq 1$. В случае, например, если $\alpha = 0$ в ячейке, то она полностью заполнена фазой 0, или, если $\alpha = 1$ в ячейке, то она полностью заполнена фазой 1.

Две несжимаемые и несмешиваемые фазы представлены в вычислительной области некоторой смесью с физическими характеристиками, посчитанными по принципу весового среднего (6), (7):

$$\rho = \rho_1 \alpha + \rho_0 (1 - \alpha), \tag{6}$$

$$\mu = \nu \rho, \quad \nu = \nu_1 \alpha + \nu_0 (1 - \alpha), \tag{7}$$

где ρ_0 и ν_0, ρ_1 и ν_1 – плотность и эффективный кинематический коэффициент вязкости каждой из фаз, соответственно.

При моделировании потоков на склонах вязкость материала склонового потока (снега, грязекаменной или водоснежной смеси) не является константой, а зависит от скорости сдвига $\dot{\gamma} = \sqrt{2\overline{\mathbf{s}} \cdot \overline{\mathbf{s}}}$:

$$\nu_1 = \nu_1(\dot{\gamma}) = \min\left(\nu_{ref}, \frac{\tau_{ref}}{\dot{\gamma}} + K\dot{\gamma}^{n-1}\right). \tag{8}$$

В последнем выражении предполагается, что фаза, представляющая материал склонового потока, рассматривается как жидкость Хершеля-Балкли. Здесь K – коэффициент консистенции, τ_{ref} – предел текучести.

Более детальное описание математической модели можно найти в книге Ферцигера (Joel H. Ferziger) и Перича (Milovan Peric) [30].

4. Вычислительный метод

Для решения вышеперечисленных систем уравнений используется свободно распространяемый пакет с открытым исходным кодом OpenFOAM. Для уравнений мелкой воды использовался решатель faSavageHutterFoam. Для решения трёхмерной задачи динамики многофазного течения используется решатель interFoam.**Модель с использованием уравнений мелкой воды**

Используются следующие аппроксимационные схемы:

- производные по времени $\frac{\partial}{\partial t}$ аппроксимируются с помощью неявного метода Эйлера;
- оператор градиента $\boldsymbol{\nabla}$ аппроксимируется центрально-разностной схемой;
- оператор дивергенции $\boldsymbol{\nabla} \cdot$ аппроксимируется с помощью противопоточной схемы с настраиваемыми весами;
- оператор Лапласа $\boldsymbol{\nabla}^2$ аппроксимируется центрально-разностной схемой с явной неортогональной коррекцией;
- другие, не перечисленные выше члены описываются с помощью центрально-разностной схемы.

В библиотеке faSavageHutterFoam реализован следующий алгоритм решения системы уравнений:

1) решается уравнение для нахождения давления на дне p_b (нормальная к склону составляющая закона сохранения количества движения);
2) решается уравнение для скорости (закон сохранения количества движения в проекции на касательную к склону плоскость), в котором используются обновленные значения давления на дне;

3) последним решается осреднённое по глубине уравнение неразрывности для нахождения глубины потока.

4.2 Модель с использованием многофазного подхода

Используются следующие аппроксимационные схемы:

- производные по времени $\frac{\partial}{\partial t}$ аппроксимируются с помощью неявного метода Эйлера;
- поток объёмной доли фазы $\nabla \cdot (\overline{\mathbf{u}}\alpha)$ аппроксимируется при помощи схемы Ван Лира;
- поток массы смеси $\nabla \cdot (\rho\overline{\mathbf{u}\mathbf{u}})$ аппроксимируется с помощью противопоточной схемы с весами;
- дивергенция тензора вязких напряжений $\nabla \cdot \overline{\boldsymbol{\tau}}$ аппроксимируется с помощью центральной разностной схемы;
- поток диссипации турбулентной кинетической энергии $\nabla \cdot (\rho\overline{\mathbf{u}}\varepsilon)$ используется противопоточная схема;
- поток турбулентной кинетической энергии $\nabla \cdot (\rho\overline{\mathbf{u}}k)$ аппроксимируется противопоточной схемой;
- оператор градиента ∇ аппроксимируется центрально-разностной схемой;
- оператор Лапласа ∇^2 аппроксимируется центрально-разностной схемой с явной неортогональной коррекцией;
- другие, не перечисленные выше члены описываются с помощью центрально-разностной схемы.

Для решения системы уравнений используется алгоритм PIMPLE, являющийся комбинацией алгоритмов PISO (Pressure Implicit with Splitting of Operator) и SIMPLE (Semi-Implicit Method for Pressure-Linked Equations).

*5. Объекты исследования*Верификационный эксперимент

Для верификации используемого программного обеспечения было решено произвести моделирование эксперимента среднего масштаба, с длиной потока порядка десятка метров. Были изучены данные трёх экспериментальных установок: установка в Давосе, Швейцария (построена в 1950 году) [31], [32], лабораторная установка Исландского университета [33], [34], установка Национального исследовательского института наук о Земле и предотвращения стихийных бедствий (NIED), Нагаока, Япония [18], [19].

Установка в Давосе в длину составляет 34 метра и 2.5 метра в ширину, для изучения динамики потоков на данной установке использовался искусственный и натуральный снег. Установка Исландского университета составляет порядка 12 метров в длину и 1 метр в ширину. На данной установке моделировался водоснежный поток с характерной плотностью потока от 800 до 1000 кг/м^3, таким образом, в эксперименте в лотке спускали воду. Экспериментальная установка Национального исследовательского института наук о Земле и предотвращения стихийных бедствий (NIED) в Японии составляет 7 метров в длину и 70 сантиметров в ширину. В качестве материала потока на данной установке используются различной плотности смеси искусственного снега и воды, а также растительное масло.

Все вышеперечисленные установки позволяют исследовать взаимодействие потока с различными препятствиями (валы, дамбы, надолбы и другие). В настоящей работе моделируется эксперимент поставленный в Университете Исландии, схема эксперимента представлена на рис. 1. В эксперименте проводилось исследование эффективности различных комбинаций и форм заградительных сооружений, например, один или два ряда надолбов и заградительная дамба разного угла наклона, или одна или две небольших прямоугольных заградительных дамбы и большая насыпь. По результатам эксперимента было выявлено, что наиболее эффективной является комбинация из двух небольших

заградительных дамб и одной основной большой заградительной дамбы, установленной вертикально.

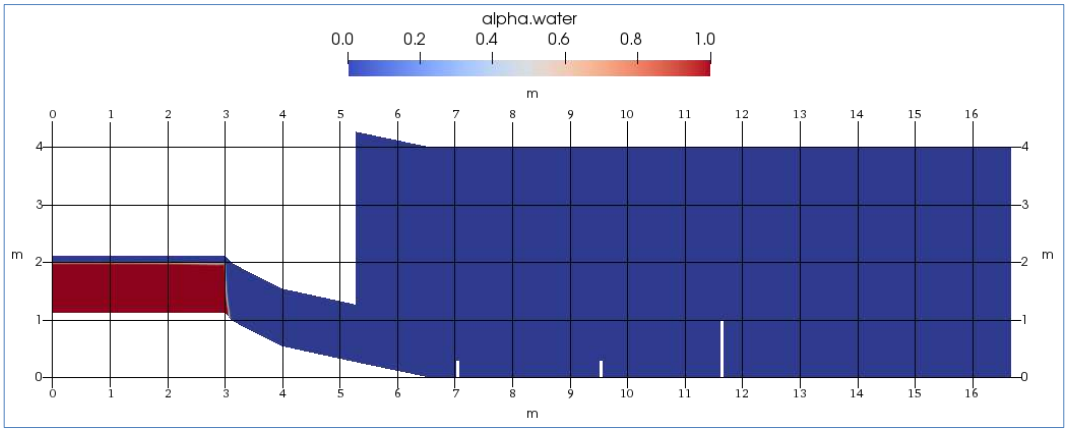

Рис. 1. Схема эксперимента: на расстоянии 7 м и 9.5 м расположены две маленькие дамбы, на расстоянии 11.6 м расположена большая дамба, красным цветом показано расположение материала потока в начальный момент времени, синим обозначен воздух
Fig. 1. Scheme of the experiment: two small dams are located at a distance of 7 m and 9.5 m, a large dam is located at a distance of 11.6 m, the location of the flow material at the initial moment of time is shown in red, air is indicated in blue

В настоящей статье рассчитывается взаимодействие потока с данной комбинацией заградительных сооружений, а также спуск потока при отсутствии заградительных сооружений. Две маленькие дамбы высотой 0.3 м расположены на расстоянии 7 м и 9.5 м от начала установки, основная заградительная дамба высотой 1 м, установленная вертикально, расположена на расстоянии 11.6 м от начала экспериментальной установки. По склону спускается 2.7 $м^3$ воды, измеряется высота всплесков на каждом из препятствий, длительность взаимодействия потока с основной дамбой (от момента встречи потока с дамбой, до установления равновесия жидкости в лотке экспериментальной установки), объём потока перелившийся через основную заградительную дамбу.

Для верификации решений на эксперименте сравним значения скорости и глубины потока, полученные из вычислительного эксперимента, с результатами натурного на расстоянии 11.1 метра от начала экспериментальной установки при отсутствии заградительных дамб.

На рис. 2 можно видеть, что расчёт с помощью трёхмерного моделирования с использованием решателя interFoam значительно лучше воспроизводит эксперимент в то время, как расчёт с использованием уравнений мелкой воды показывает значительное опережение реального фронта вычисленным, глубина потока занижена по сравнению с экспериментом, а скорость завышена (кроме скорости фронтальной части потока).

При расчёте взаимодействия потока с двумя маленькими дамбами и одной большой, процесс делится на три фазы: фаза начального всплеска на основной дамбе (большой), квазистационарная фаза, когда происходит переливание потока через основную дамбу, после окончания этой фазы наступает третья фаза – до момента установления равновесия жидкости в экспериментальной установке. Расчет проводится по трехмерной модели, для этих трёх фаз сравниваются параметры потока, приведенные в табл. 1.

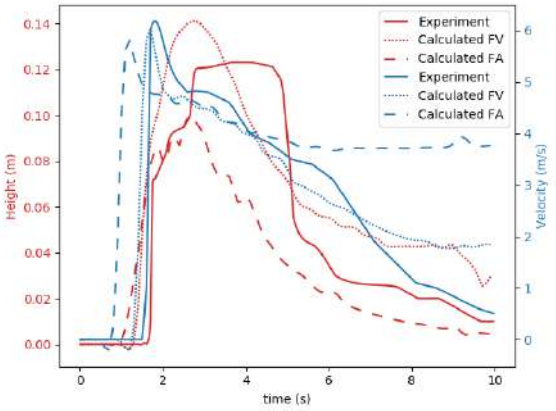

Рис. 2. Графики скорости (синий) и глубины потока (красный), замеренные на расстоянии 11.1 метра от начала установки для натурного эксперимента (Experiment), вычисленные с помощью решателя faSavageHutterFoam (Calculated FA) и вычисленные с помощью решателя interFoam (Calculated FV)
Fig. 2. Flow velocity (blue) and flow depth (red) plots measured at a distance of 11.1 meters from the start of the setup for an experiment (Experiment), calculated data using the faSavageHutterFoam solver (Calculated FA) and calculated data using the interFoam solver (Calculated FV)

Табл. 1. Сравнение измеренных и рассчитанных параметров потока
Table 1. Comparison of measured and calculated flow parameters

Параметр сравнения	Экспериментальные данные	Вычисленные данные
Высота первого всплеска на основной дамбе	1.3 м	1.45 м
Высота потока, переливающегося через дамбу	0.5 м	0.05 м
Средняя высота потока перед основной дамбой в третьей фазе	0.4 м	0.45 м
Время с начала взаимодействия потока с основной дамбой, до окончания переливания потока через дамбу	1.25 с	0.9 с
Объём перелившейся через основную дамбу	2.684 $м^3$	2.133 $м^3$

По вышеперечисленным результатам можно сделать следующие выводы: расчёт высоты первоначального всплеска отличается от экспериментального на 10%, в параметрах высоты потока, переливающегося через дамбу и времени взаимодействия потока с дамбой расчёт переоценивает эффективность работы дамбы, таким образом вычисленный объём потока, перепрыгнувшего через дамбу, не совпадает с измеренным. Высота потока при распространении возмущений вверх от основной дамбы даёт хорошее совпадение с экспериментом.

5.2 Натурный склон

В качестве расчётной области взят участок склона горы Юкспор (Хибины), включающий 22 лавинный очаг (цифровая модель рельефа предоставлена научно-исследовательской лабораторией снежных лавин и селей). Для модели с использованием уравнений мелкой воды расчётная сетка плоская, задаётся с использованием цифровой модели рельефа склона. Для модели с использованием многофазного подхода – это трёхмерная область, верхняя и нижняя границы которой коллинеарны склону, по толщине сетка содержит много слоёв. На рис. 3 показана цифровая модель рельефа 22 лавинного очага (правое русло), голубым цветом отмечена зона зарождения лавины, зелёным – зона лавинных отложений, конусы 1, 2, 3 показывают точки замера высоты и скорости потока в численном эксперименте.

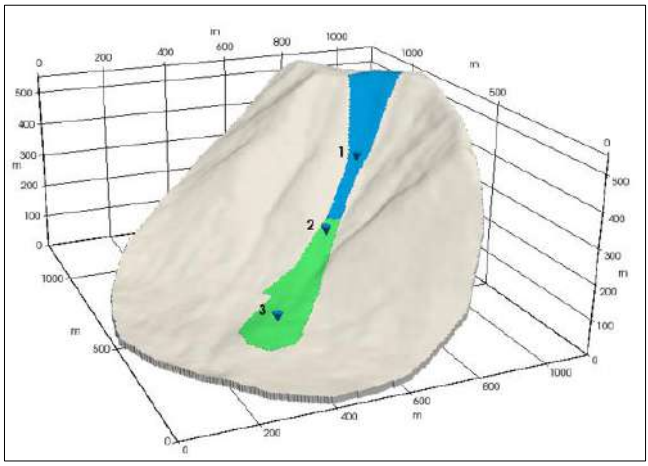

Рис. 3. Цифровая модель рельефа 22 лавинного очага с точками замера глубины и скорости потока и зонами зарождения лавины (голубой) и лавинных отложений (зелёный)
Fig. 3. Digital elevation model of 22 avalanche area with measurement points for flow depth and flow velocity and zones of avalanche start area (blue) and avalanche deposits (green)

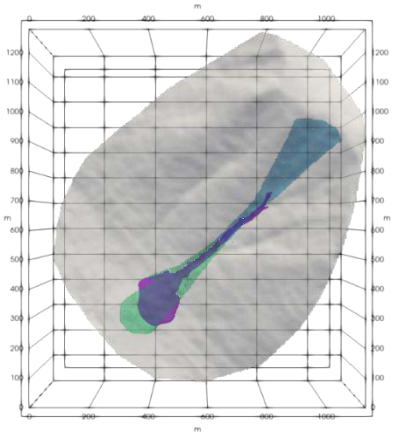

Рис. 4. Расчёт методом конечных площадок с использованием уравнений, осреднённых по глубине. Фиолетовым цветом отмечена расчётная зона лавинных отложений
Fig. 4. Finite area method calculation using depth-averaged equations. The calculated zone of avalanche deposits is marked in purple

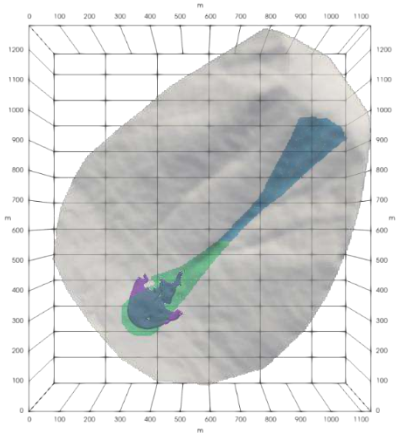

Рис. 5. Расчёт методом конечных объёмов с использованием многофазной модели потока. Фиолетовым цветом отмечена расчётная зона лавинных отложений
Fig. 5. Finite volume method calculation using a multiphase flow model. The calculated zone of avalanche deposits is marked in purple

6. Начальные и граничные условия В начальный момент времени задан неподвижный слой снега толщиной 2 метра в зоне зарождения, остальная часть расчётной области заполнена воздухом, который тоже неподвижен.

На верхней и боковых границах расчётной области задано условие нулевого градиента. На нижней границе расчётной области, являющейся твёрдой поверхностью склона, задано условие прилипания.

7. Результаты

При сравнении дальности выброса и формы лавинных отложений для двух подходов, описанных в данной работе, были получены следующие результаты.

На рис. 4 и 5 видно достаточно хорошее совпадение формы лавинных отложений с натурными данными. При расчёте с использованием многофазного подхода дальность выброса лавины хорошо совпадает с натурными данными, форма лавинных отложений при расчёте с помощью уравнений мелкой воды больше повторяет контуры реальных лавинных отложений.

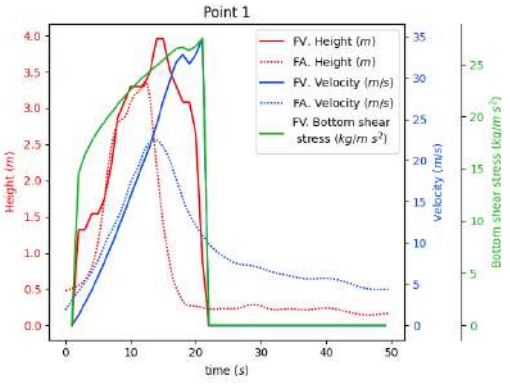

Рис. 6. Изменение глубины, средней скорости потока, напряжения на дне со временем в точке 1

Fig. 6. Dynamic of flow depth, average flow rate, bottom stress at point 1

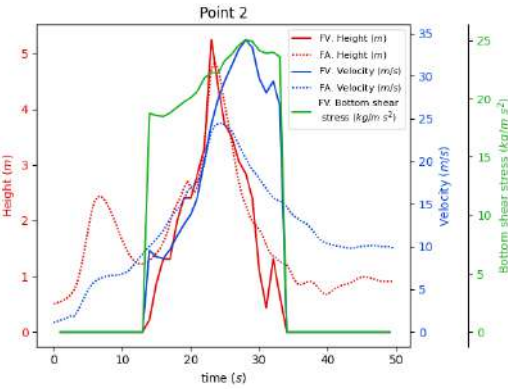

Рис. 7. Изменение глубины, средней скорости потока, напряжения на дне со временем в точке 2

Fig. 7. Dynamic of flow depth, average flow rate, bottom stress at point 2

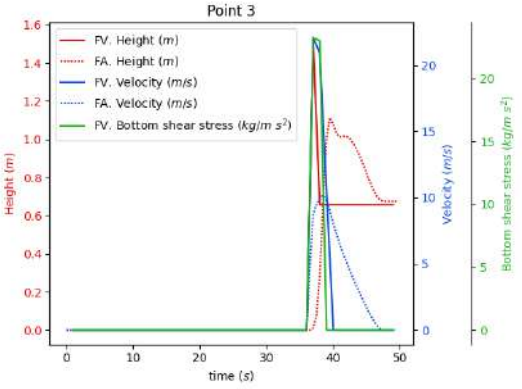

Рис. 8. Изменение глубины, средней скорости потока, напряжения на дне со временем в точке 3

Fig. 8. Dynamic of flow depth, average flow rate, bottom stress at point 3

В результате расчётов были получены графики глубины потока, скорости потока, а также значения напряжения на дне показанные на рис. 6-8. Скорость потока в расчёте с помощью многофазного подхода (FV) принимается равной нулю при значении, меньшем, чем 10^{-6} м/с. Последние данные особенно важны для разработки модели захвата подстилающего материала потоком, так как основным критерием разрушения подстилающего материала является превышение напряжения на дне некоторого порогового значения. Исследования различных механизмов разрушения потоком подстилающей поверхности и уноса материала представлены в работах работах Эглит и Якубенко [35], [36], Эглит и Демидова [37], Исслера

(Dieter Issler) и соавторов [38], [39], Бейтс (Belinda M. Bates), Андрени (Nicolas Andreini) и Ансея (Christophe Ancey) [40], Фишера (Jan-Thomas Fischer) и соавторов [26] и других.

Для уточнения модели, описывающей трение на дне, в подходе с использованием уравнений типа мелкой воды можно сделать следующие выводы по результатам проведённых исследований (исходя из данных трёхмерного моделирования). Касательное напряжение на дне растёт вместе с ростом скорости и глубины потока до некоторого порогового значения – данный рост можно наблюдать в точке 1 на рис. 6. Ближе к 17 секунде движения потока при увеличении глубины и скорости потока рост сдвигового напряжения на дне замедляется. Далее в точке 2, находящейся в зоне транзита (рис. 7), сдвиговое напряжение ниже, чем в точке 1, несмотря на то, что глубина и скорость потока в точке 2 выше, чем в точке 1. Такая зависимость сдвигового напряжения от скорости и глубины потока говорит о присутствии порогового значения для величины трения, описанного Григоряном в работе [41]. То есть при малой толщине потока напряжение сдвига на дне растёт с ростом мощности (толщины) потока, достигая некоторой предельной величины, определяемой эффективной сдвиговой прочностью. С дальнейшим ростом мощности (толщины) потока напряжение сдвига на дне остаётся на этом постоянном уровне, тогда как движущая сила пропорциональная глубине потока, растёт с увеличением толщины потока.

8. Разработка прототипа модуля для расчёта захвата донного материала и отложения материала потока на склоне для трёхмерной многофазной модели

Для учёта явления захвата донного материала и отложения материала потока на склоне, удобно ввести дополнительную фазу: неподвижный материал склона, который может быть вовлечён в движение. Данная фаза обладает свойствами, отличными от свойств движущегося материала потока и воздуха. Также необходимо задать условие фазового перехода материала склона в материал потока при захвате, и обратного фазового перехода при отложении. Условием возникновения фазового перехода при разрушении подстилающего материала является превышение сдвиговыми напряжениями разрушающего порогового значения при условии, что скорости деформаций в данной точке подстилающего материала отличны от нуля. Обратный переход отложения материала потока в материал склона происходит при условии уменьшения сдвиговых напряжений до порогового значения отложения. Последнее условие применяется на границе материала склона и материала потока, где скорости деформации среды максимальные по глубине потока.

Введённая новая фаза подстилающего материла неподвижна до момента разрушения слоя потоком. В случае использования многофазной односкоростной модели, аналогичной математической модели, представленной в разделе 4.2, линейный размер ячейки должен быть меньше предполагаемой глубины разрушения подстилающего материала за шаг по времени. В расчётах, проведённых для натурного склона, шаг по времени составлял в среднем 10^{-2} секунды. Исходя из оценки скорости захвата проведённой Эглит и Якубенко [35] линейный размер ячейки должен быть не более 10^{-3}. Такое разрешение сетки невозможно удовлетворить в силу недостаточности вычислительных ресурсов. Таким образом, необходимо вводить многоскоростную многофазную модель.

Для получившейся трёхфазной модели верны следующие законы переноса объёмной доли фаз (a (air) – воздух, f (flow) – материал движущегося потока, s (slope) – материал склона или материал подстилающей поверхности):

$$\frac{\partial \alpha_a}{\partial t} + \nabla \cdot (\mathbf{u}_a \alpha_a) = q_{f\to a} - q_{a\to f} + q_{s\to a} - q_{a\to s},$$
$$\frac{\partial \alpha_f}{\partial t} + \nabla \cdot (\mathbf{u}_f \alpha_f) = q_{a\to f} - q_{f\to a} + q_{s\to f} - q_{f\to s},$$
$$\frac{\partial \alpha_s}{\partial t} + \nabla \cdot (\mathbf{u}_s \alpha_s) = q_{a\to s} - q_{s\to a} + q_{f\to s} - q_{s\to f}.$$

Здесь $\mathbf{u}_k$ – скорость k-ой фазы; α_k – объёмная доля k-ой фазы; $q_{k\to n}$ – фазовый переход из фазы k в фазу n.

Для каждой из трёх фаз рассчитывается закон сохранения количества движения:

$$\frac{\partial (\alpha_a \rho_a \mathbf{u}_a)}{\partial t} + \nabla \cdot (\alpha_a \rho_a \mathbf{u}_a \mathbf{u}_a) = -\alpha_a \nabla p + \nabla \cdot \boldsymbol{\tau}_a + \rho \mathbf{f}_a + \mathbf{M}_{d,a},$$
$$\frac{\partial (\alpha_f \rho_f \mathbf{u}_f)}{\partial t} + \nabla \cdot (\alpha_f \rho_f \mathbf{u}_f \mathbf{u}_f) = -\alpha_f \nabla p + \nabla \cdot \boldsymbol{\tau}_f + \rho \mathbf{f}_f + \mathbf{M}_{d,f},$$
$$\frac{\partial (\alpha_s \rho_s \mathbf{u}_s)}{\partial t} + \nabla \cdot (\alpha_s \rho_s \mathbf{u}_s \mathbf{u}_s) = -\alpha_s \nabla p + \nabla \cdot \boldsymbol{\tau}_s + \rho \mathbf{f}_s + \mathbf{M}_{d,s}.$$

Здесь $\boldsymbol{\tau}_k$ – тензор напряжений k-ой фазы; ρ_k – плотность k-ой фазы; p – давление смеси; $\mathbf{f}_k$ – плотность массовых сил для k-ой фазы; $M_{d,k}$ – обмен импульсом между фазами за счёт трения.

Предлагается использовать следующие замыкающие соотношения для трёх фаз (воздух, движущийся материал потока, покоящийся материала склона), представленные в таблице 2

Табл. 2. Реологические свойства и фазовые переходы для трёхфазного потока
Table 2. Rheological properties and phase transitions for three-phase flow

alpha.air	**alpha.flow**	**alpha.slope**
Ньютоновская среда	**Среда Хершеля–Балкли**	**Покоящаяся среда**
$\boldsymbol{\tau}_a = 2\mu_a \mathbf{s}_a$	$\boldsymbol{\tau}_f = \tau_{f0} + K_f (I_2(\mathbf{s}_f))\mathbf{s}_f$, $K_f = 2\mu_{f_0} I_2(\mathbf{s}_f)^{n-1}$, $0.3 < n_f < 0.8$	$\boldsymbol{\tau}_s = \tau_{s0} + 2\mu_s \mathbf{s}_s$, τ_{s0} – большое число, $\mu_s \to \inf$
$q_{f\to a} = q_{a\to f} = q_{s\to a} = q_{a\to s} = 0,$ $q_{s\to f} = \lambda_{s\to f}\alpha_s$, если $I_2(\boldsymbol{\tau}_s) > \tau_{sr}$, и $\alpha_s > \alpha_{s0}$, $q_{f\to s} = \lambda_{f\to s}\alpha_f$, если $I_2(\mathbf{s}_f)/\lvert\mathbf{u}_f\rvert > c_{fs}$, и $I_2(\boldsymbol{\tau}_f) < \tau_{fr}$		
$\mathbf{M}_{d,a} = \mathbf{M}_{d,f\to a} - \mathbf{M}_{d,a\to f} + \mathbf{M}_{d,s\to a} - \mathbf{M}_{d,a\to s}$	$\mathbf{M}_{d,f} = \mathbf{M}_{d,f\to a} - \mathbf{M}_{d,a\to f} + \mathbf{M}_{d,s\to f} - \mathbf{M}_{d,f\to s}$	$\mathbf{M}_{d,s} = \mathbf{M}_{d,f\to s} - \mathbf{M}_{d,s\to f} + \mathbf{M}_{d,s\to a} - \mathbf{M}_{d,a\to s}$
$\mathbf{M}_{d,f\to a} = C_{d,f\to a}(\mathbf{u}_f - \mathbf{u}_a)$, $\mathbf{M}_{d,a\to f} = C_{d,a\to f}(\mathbf{u}_a - \mathbf{u}_f)$, $\mathbf{M}_{d,s\to a} = C_{d,s\to a}(\mathbf{u}_s - \mathbf{u}_a)$, $\mathbf{M}_{d,a\to s} = C_{d,a\to s}(\mathbf{u}_a - \mathbf{u}_s)$, $\mathbf{M}_{d,s\to f} = C_{d,s\to f}(\mathbf{u}_s - \mathbf{u}_f)$, $\mathbf{M}_{d,f\to s} = C_{d,f\to s}(\mathbf{u}_f - \mathbf{u}_s)$		

Здесь $\lambda_{s\to f}$ – скорость захвата материала подстилающей поверхности, $\lambda_{f\to s}$ – скорость отложения материала потока; $C_{d,k\to n}$ – скорость передачи импульса фазой k фазе n за счёт сил трения.

Для реализации предложенной модели в пакете OpenFOAM, предполагается создать новый решатель multiphaseEulerChangeFoam на базе multiphaseEulerFoam [42], с добавлением фазового перехода [43]. Так же предполагается создание новых моделей фазовых переходов.

Код решателя multiphaseEulerFoam располагается в директории `$FOAM_INSTALL_DIR/applications/solvers/multiphase/multiphaseEulerFoam`.

Файл `multiphaseEulerFoam.C` является основным файлом, содержащим в себе вызовы подключаемых модулей, как например модели фаз (`phaseModel.H` описывает свойства среды, такие как плотность, вязкость, поверхностное натяжение и другие, описанные в файле

`transportProperites`), файл `multiphaseSystem.H` уписывает процедуру решения уравнений транспорта объёмных долей фаз, файл `dragModel.H` содержит в себе различные модели межфазного трения, файл `turbulentTransportModel.H` описывает модели турбулентности.

Выше были перечислены основные подключаемые модули. Уравнения многофазной модели (уравнения переноса объёмной доли фазы, законы сохранения количества движения для каждой из фаз, уравнения турбулентной модели) решаются раздельно с помощью алгоритма PIMPLE:

- решаются уравнения турбулентной модели и корректируется поле скорости,
- решаются уравнения переноса фаз,
- рассчитывается плотность смеси,
- рассчитывается закон сохранения количества движения для каждой из фаз,
- корректируется давление.

Для реализации межфазных переходов необходимо создать класс межфазных переходов в файле `phaseChangeModel.H`, содержащий, все необходимые модели фазовых переходов.

Далее необходимо реализовать таблицу межфазных переходов, соотносящую пару фаз с межфазным переходом, используемым для данной пары.

Далее источниковый член добавляется в алгоритм решения уравнения транспорта объёмной доли, описанный в файле `multiphaseSystem.C`, в соответствии с таблицей межфазных переходов.

9. Выводы

В настоящей работе проводится моделирование эксперимента со спуском потока в лотке при наличии комплекса заградительных сооружений. Сравниваются результаты двух подходов: с использованием уравнений мелкой воды и трёхмерного многофазного подхода. Данные вычисления позволяют оценить область применимости каждой из вышеперечисленных моделей. Решение, полученное с помощью решателя interFoam, позволяет рассчитать такие параметры, как скорость потока, плотность, глубина, вязкость, сдвиговое напряжение на дне, все эти параметры важны для расчёта эффективных заградительных сооружений и позволяют заменить более дорогостоящие и трудоёмкие натурные эксперименты численными. Также трёхмерный расчёт даёт возможность точного моделирования разрушения и уноса подстилающего материала, так как известны все параметры потока на дне.

Вместе с этим, расчёт крупномасштабного склонового потока требует очень больших вычислительных ресурсов. В данной ситуации эффективно комбинировать трёхмерный многофазный подход с подходом, использующим уравнения мелкой воды, который является вычислительно менее затратным. Однако такие члены, как трение на дне и захват материала потоком, являющиеся наиболее важными для точного моделирования таких явлений, как снежные лавины или грязекаменные сели, требуют уточнения в подходе с использованием уравнений мелкой воды. Уточнения таких моделей можно получить из анализа трёхмерных расчётов, полученных в данной работе.

10. Дополнение

В работе использовались значения коэффициентов, указанные в табл. 3.

Табл. 3. Значения коэффициентов
Table 3. Coefficient values

Модель с использованием многофазного подхода	Модель с использованием уравнений мелкой воды
$\rho_{snow} = 200$ кг/м^3,	$\rho = 200$ кг/м^3,

$\rho_{air} = 1$ кг/м^3,	$\mu = 0.577$,
$\nu_{air} = 1.48 \cdot 10^{-5}$ м^2/с,	$\xi = 10^4$ м/с^2,
$\nu_{ref} = 10^7$ м^2/с,	$u_0 = 10^{-3}$ м/с.
$\tau_{ref} = 10$ кг /м с^2,	
$K = 6$ м^2/с,	
$n = 0.4$,	
$C_\mu = 0.09$,	
$C_{\varepsilon 1} = 1.44$,	
$C_{\varepsilon 2} = 1.92$,	
$\sigma_k = 1.0$,	
$\sigma_\varepsilon = 1.3$.	

Список литературы / References

[1] М.Д. Докукин, С.С. Черноморец, Е.А. Савернюк, Э.В. Запорожченко, Р.А. Бобов, У.Р. Пирмамадов. Барсемская селевая катастрофа на Памире в 2015 году и ее аналоги на Центральном Кавказе. Геориск, том 13, no. 1, 2019 г., стр. 26-36, 2019 / M.D. Dokukin, S.S. Chernomorets, E.A. Savernyuk, E.V. Zaporozhchenko, R.A. Bobov, U.R. Pirmamadov. Barsem debris flow disaster in the Pamirs in 2015 and its analogues in the Central Caucasus, Georisk, vol. 13, no. 1, 2019, pp. 26-36 (in Russian).

[2] С.С. Черноморец, Д.А. Петраков и др. Прорыв озера Башкара (Центральный Кавказ, Россия) 1 сентября 2017 года. Криосфера Земли, том. 22, no. 2, 2018 г., стр. 70-80 / S.S. Chernomorets, D.A. Petrakov et al. The outburst of Bashkara glacier lake (Central Caucasus, Russia) on September 1, 2017. Kriosfera Zemli, vol. 22, no. 2, 2018, pp. 70-80 (in Russian).

[3] M. Eglit. Some mathematical models of snow avalanches. In Advances in the Mechanics and the Flow of Granular Materials, vol. 2, Trans Tech Pubn, 1983, pp. 557-588.

[4] M. Eglit, A. Yakubenko, and J. Zayko. A review of russian snow avalanche models–from analytical solutions to novel 3D models. Geosciences, vol. 10, no. 2, 2020, article no. 77.

[5] O. Hungr. A model for the runout analysis of rapid flow slides, debris flows, and avalanches. Canadian Geotechnical Journal, vol. 32, no. 4, 1995, pp. 610–623.

[6] P. Sampl and M. Granig. Avalanche simulation with SAMOS-AT. In Proc. of the International Snow Science Workshop, 2009, pp. 519-523.

[7] P. Sampl and T. Zwinger. Avalanche simulation with SAMOS. Annals of Glaciology, vol. 38, 2004, pp. 393-398.

[8] V. Medina, M. Hurlimann, and A. Bateman. Application of flatmodel, a 2D finite volume code, to debris flows in the northeastern part of the Iberian Peninsula. Landslides, vol. 5, 2007, pp. 127-142.

[9] M. Christen, J. Kowalski, and P. Bartelt. RAMMS: Numerical simulation of dense snow avalanches in three-dimensional terrain. Cold Regions Science and Technology, vol. 63, no. 1, 2010, pp. 1-14.

[10] E.B. Pitman, C.C. Nichita, A. Patra, A. Bauer, M. Sheridan, and M. Bursik. Computing granular avalanches and landslides. Physics of Fluids, vol. 15, no. 12, 2003, pp. 3638-3646.

[11] A.K. Patra, A.C. Bauer et al. Parallel adaptive numerical simulation of dry avalanches over natural terrain. Journal of Volcanology and Geothermal Research, vol. 139, no. 1, 2005, pp. 1-21.

[12] M. Mergili, K. Schratz, A. Ostermann, and W. Fellin. Physically-based modelling of granular flows with Open Source GIS. Natural Hazards and Earth System Sciences, vol. 12, no. 1, 2012, pp. 187-200.

[13] M. Mergili, J.-T. Fischer, J. Krenn, and S. P. Pudasaini. R.avaflow v1, an advanced open-source computational framework for the propagation and interaction of two-phase mass flows. Geoscientific Model Development, vol. 10, no. 2, 2017, pp. 553-569.

[14] S. Hergarten and J. Robl. Modelling rapid mass movements using the shallow water equations in cartesian coordinates. Natural Hazards and Earth System Sciences, vol. 15, no. 3, 2015, pp. 671-685.

[15] M. Rauter and Z. Tukovic. A finite area scheme for shallow granular flows on three-dimensional surfaces. Computers & Fluids, vol. 166, 2018, pp. 184-199.

[16] M. Rauter, A. Kofler, A. Huber, and W. Fellin. FaSavageHutterFOAM 1.0: Depth-integrated simulation of dense snow avalanches on natural terrain with openfoam. Geoscientific Model Development, vol. 11, no. 7, 2018, pp. 2923-2939.

[17] H. Orn Petursson, K. M. Hakonardottir, and A. Thoroddsen. Use of OpenFOAM and RAMMS Avalanche to simulate the interaction of avalanches and slush flows with dams. In Proc. of the International Symposium on Mitigative Measures against Snow Avalanches and Other Rapid Gravity Mass Flows, 2019, pp. 1-12.

[18] Y. Yamaguchi, S. Takase, S. Moriguchi, K. Terada, K. Oda, and I. Kamiishi. Three-dimensional nonstructural finite element analysis of snow avalanche using non-newtonian fluid model. Transactions of the Japan Society for Computational Engineering and Science, vol. 2017, paper no. 20170011 (in Japanese).

[19] K. Oda, S. Moriguchi, I. Kamiishi, A. Yashima, K. Sawada, and A. Sato. Simulation of a snow avalanche model test using computational fluid dynamics. Annals of Glaciology, vol. 52, no. 58, 2011, pp. 57-64.

[20] Романова Д.И. Трёхмерное моделирование потоков жидкости Хершеля-Балкли на склоне в OpenFOAM. Труды ИСП РАН, том 29, вып. 1, 2017 г., стр. 85-100 / Romanova D.I. 3D flow modeling of Herschel-Bulkley fluid on the slope in OpenFOAM. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 1, 2017, pp. 85-100 (in Russian). DOI: 10.15514/ISPRAS-2017-29(1)-6.

[21] С.С. Григорян, М.Э. Эглит, Ю.Л. Якимов. Новая математическая постановка задачи о движении лавины и решение этой задачи. Труды Высокогорного геофизического института, no 12, 1967 г., стр. 104-113 / S.S. Grigorian, M.E. Eglit, and Y.L. Iakimov. A new formulation and solution of the problem of snow avalanche motion. Trudy Vycokogornogo Geofizicheskogo Instituta, no. 12, 1967, pp. 104-113 (in Russian).

[22] S.B. Savage and K. Hutter. The motion of a finite mass of granular material down a rough incline. Journal of Fluid Mechanics, vol. 199, 1989, pp. 177-215.

[23] S.B. Savage and K. Hutter. The dynamics of avalanches of granular materials from initiation to runout. Part I: Analysis. Acta Mechanica, vol. 86, no. 1, 1991, pp. 201-223.

[24] R. Greve, T. Koch, and K. Hutter. Unconfined flow of granular avalanches along a partly curved surface. I. Theory. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, vol. 445, no. 1924, 1994, pp. 399-413.

[25] A. Voellmy and A. Roch, Über die zerstörungskraft von lawinen. Schweizerische Bauzeitung, 73, 1955 (in German).

[26] J.-T. Fischer, A. Kofler, W. Fellin, M. Granig, and K. Kleemayr. Multivariate parameter optimization for computational snow avalanche simulation. Journal of Glaciology, vol. 61, no. 229, 2015, pp. 875-888.

[27] S.H.E. Tahry. K-epsilon equation for compressible reciprocating engine flows. Journal of Energy, vol. 7, no. 4, 1983, pp. 345-353.

[28] B. Launder and D. Spalding. The numerical computation of turbulent flows. Computer Methods in Applied Mechanics and Engineering, vol. 3, no. 2, 1974, pp. 456-460.

[29] C. Hirt and B. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. Journal of Computational Physics, vol. 39, no. 1, 1981, pp. 201-225.

[30] J. Ferziger and M. Peric. Computational methods for fluid dynamics, 3rd edition. Springer, 2002, 426 p.

[31] K. Hakonardottir, A. Hogg, T. Johannesson, M. Kern, and F. Tiefenbacher. Large-scale avalanche braking mound and catching dam experiments with snow: A study of the airborne jet. Surveys in Geophysics, vol. 24, no. 5-6, 2003, pp. 543-554.

[32] C. Jaedicke, M. Kern, P. Gauer, M.-A. Baillifard, and K. Platzer. Chute experiments on slushflow dynamics. In Proc. of the 2006 International Snow Science Workshop, 2006, pp. 139-147.

[33] K.H. Agustsdottir. The design of slushflow barriers: Laboratory experiments. PhD thesis, Faculty of Industrial Engineering, Mechanical Engineering, and Computer Science, University of Iceland, 2019, 58 p.

[34] R.A. Jones. The design of slushflow barriers: CFD simulations. PhD thesis, Faculty of Industrial Engineering, Mechanical Engineering, and Computer Science, University of Iceland, 2019, 52 p.

[35] M. Eglit and A. Yakubenko. Numerical modeling of slope flows entraining bottom material. Cold Regions Science and Technology, vol. 108, 2014, pp. 139-148.

[36] М.Э. Эглит, А.Е. Якубенко. Влияние захвата донного материала и неньютоновской реологии на динамику турбулентных склоновых потоков. Известия Российской академии наук. Механика

жидкости и газа, no. 3, 2016 г., pp. 3-15 / M.E. Eglit, A.E. Yakubenko, Effect of the bottom material capture and the non-newtonian rheology on the dynamics of turbulent downslope flows. Fluid Dynamics, vol. 51, no. 3, 2016, pp. 299-310.

[37] M. Eglit and K. Demidov. Mathematical modeling of snow entrainment in avalanche motion. Cold Regions Science and Technology, vol. 43, no. 1, 2005, pp. 10-23.

[38] D. Issler. Dynamically consistent entrainment laws for depth-averaged avalanche models. Journal of Fluid Mechanics, vol. 759, 2014, pp. 701-738.

[39] D. Issler and M. Pastor Pérez. Interplay of entrainment and rheology in snow avalanches: A numerical study. Annals of Glaciology, vol. 52, no. 58, 2011, pp. 143-147.

[40] B. Bates, N. Andreini, and C. Ancey. Basal entrainment by newtonian gravity-driven flows. Physics of Fluids, vol. 28, no. 5, 2016, article no. 053101.

[41] С.С. Григорян. Новый закон трения и механизм крупномасштабных горных обвалов и оползней. Доклады Академии наук СССР, том 244, no. 4, 1979 г., pp. 846-846 / S.S. Grigoryan. A new friction law and mechanism of large-scale rock falls and mountain creeps. Doklady Akademii Nauk SSSR, vol. 244, no. 4, 1979, pp. 846-849 (in Russian).

[42] K. Wardle and H. Weller. Hybrid multiphase CFD solver for coupled dispersed/segregated flows in liquid-liquid extraction. International Journal of Chemical Engineering, vol. 2013, 2013, article ID 128936, 13p.

[43] V.K. Oruganti. Implementation of cavitation models into the multiphaseEulerFoam solver. In Proc. of CFD with OpenSource Software, 2017, 50 p.

Информация об авторе / Information about the author

Дарья Игоревна РОМАНОВА получила степень магистра на кафедре гидромеханики механико-математического факультета МГУ в 2017 году, в настоящее является младшим научным сотрудником лаборатории вычислительных методов механико-математического факультета МГУ и стажёром-исследователем ИСП РАН. Она является разработчиком программного обеспечения с открытым исходным кодом для численного моделирования задач механики сплошных сред, включая турбулентные течения, течения со свободной поверхностью, многофазные течения с фазовым переходом и включениями твердых частиц.

Daria Igorevna ROMANOVA received her master's degree at the Department of Hydromechanics, Faculty of Mechanics and Mathematics, Moscow State University in 2017, and is currently a junior researcher at the Laboratory of Computational Methods at the Faculty of Mechanics and Mathematics of Moscow State University and an intern-researcher at ISP RAS. She is an open source software developer for the numerical modeling of continuum mechanics problems, including turbulent flows, free surface flows, multiphase flows with a phase transition, and solid inclusions.

DOI: 10.15514/ISPRAS–2020–32(6)–15

Моделирование аккумуляции кинетической энергии внутренних волн в областях с большим отношением горизонтального и вертикального масштабов

[1] *С.А. Елистратов, ORCID: 0000-0002-7006-6879 <sa.elist-ratov@yandex.ru>*
[1,2] *К.А. Ватутин, ORCID: 0000-0002-4090-6320 <vatutin.k@mail.ru>*
[1,2,3] *И.Н. Сибатуллин, ORCID: 0000-0003-2265-3259 <sibgat@ocean.ru>*
[4] *Е.В. Ерманюк, ORCID: 0000-0002-9989-8222 <ermanyuk@gmail.com>*
[1] *Е.А. Михайлов, ORCID: 0000-0002-9747-4039 <ea.mikhajlov@physics.msu.ru>*

[1] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1.*
[2] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[3] *Институт океанологии им. П.П. Ширшова РАН, 109004, Россия, г. Москва, Нахимовский проспект, д. 36*
[4] *Институт гидродинамики им. М.А. Лаврентьева, 630090, Россия, Новосибирск, пр. Лаврентьева, 15*

Аннотация. Под воздействием приливных явлений в глубине океана возникают внутренние волны. Глубоководный океан является примером непрерывной плотностной стратификации в жидкости, подверженной крупномасштабным периодическим возмущениям. При наличии наклонных границ, в силу особенностей дисперсионного соотношения, в такой среде могут возникать геометрические области притяжения волновых пучков после последовательных отражений от границ. Ранее особенности поведения кинетической энергии при возникновении волновых аттракторов исследовались для двумерных областей с сопоставимыми масштабами по вертикали и горизонтали. При увеличении аспектного соотношения концентрация кинетической энергии может качественно изменяться как в ламинарном, так и в турбулентном режиме. В настоящей работе показано, что концентрация удельной кинетической энергии может на порядки возрастать при увеличении аспектного соотношения. Источником возмущений являются крупномасштабные монохроматические колебания верхней горизонтальной границы области. Проведено численное моделирование эволюции волновых движений в удлиненной по горизонтали области при различных значениях амплитуды возмущения, выявлены принципиальные особенности этого случая по сравнению со случаем области, имеющей соизмеримые значения глубины и длины. Рассмотрено несколько наиболее характерных ситуаций, изучены интегральные механические характеристики: полная диссипация, средняя кинетическая энергия и ее пульсации в ламинарных и турбулентных режимах.

Ключевые слова: внутренние волны; волновая турбулентность; волновой аттрактор; неустойчивость

Для цитирования: Елистратов С.А., Ватутин К.А., Сибатуллин И.Н., Ерманюк Е.В., Михайлов Е.А. Моделирование аккумуляции кинетической энергии внутренних волн в областях с большим соотношением горизонтального и вертикального размеров. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 201-212. DOI: 10.15514/ISPRAS–2020–32(6)–15

Благодарности: Работа выполнена при финансовой поддержке Российского научного фонда (РНФ), грант 19-11-00169 с использованием оборудования Центра коллективного пользования

сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова и вычислительного комплекса Центра коллективного пользования НИЦ Курчатовский институт.

Numerical Simulation of Internal Waves and Effects of Accumulation of Kinetic Energy in Large Aspect Ratio Domains

[1] *S.A. Elistratov, ORCID: 0000-0002-7006-6879 <invsbl_mn@mail.ru>*
[1,2] *K.A. Vatutin, ORCID: 0000-0002-4090-6320 <vatutin.k@mail.ru>*
[1,2,3] *I.N. Sibgatullin, ORCID: 0000-0003-2265-3259 <sibgat@ocean.ru>*
[4]*E.V. Ermanyuk , ORCID*: *0000-0002-9989-8222 <ermanyuk@gmail.com>*
[1] *E.A. Mikhajlov, ORCID: 0000-0002-9747-4039 <ea.mikhajlov@physics.msu.ru>*
[1] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[3] *Shirshov Oceanology Institute of Russian Academy of Sciences,*
36, Nakhimovsky prospect, Moscow, Russia.
[4]*Lavrentyev Institute of Hydrodynamics*
15, Lavrentiev prospect, Novosibirsk, 630090, Russia

Abstract. Tidal forcing excites internal waves in the bulk of the ocean. Deep ocean is an example of a system with continuous stratification subject to large-scale periodic forcing. Owing to specific dispersion relation of internal waves, the domains bounded by sloping boundaries may support wave patterns with wave rays converging to closed trajectories (geometric attractors) as result of iterative focusing reflections. Previously the behavior of kinetic energy in wave attractors has been investigated in two-dimensional domain with comparable depth and length. As the geometric aspect ratio of the domain increases, the dynamic pattern of energy focusing may significantly evolve both in laminar and turbulent regimes. The present paper shows that the energy density in domains with large aspect ratio can significantly increase. In numerical simulations the input forcing has been introduced at global scale by prescribing small-amplitude deformations of the upper bound of the liquid domain. The evolution of internal wave motion in such system has been computed numerically for different values of the forcing amplitude. The behavior of the large-aspect-ratio system has been compared to the well-studied case of the system with depth-to-length ratio of order unity. A number of most typical situations has been analysed in terms of behavior of integral mechanical quantities such as total dissipation, mean kinetic energy and energy fluctuations in laminar and turbulent cases.

Keywords: internal waves; wave turbulence; wave attractor; instability

For citation: Elistratov S.A., Vatutin K.A., Sibgatullin I.N., Ermanyuk E.V., Mikhajlov E.A. Numerical simulation of internal waves and effects of accumulation of kinetic energy in large aspect ratio domains. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 201-212 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–15

Acknowledgements. This work was carried out with financial support from the Russian Science Foundation (RSF), grant 19-11-00169, using the HPC computing resources at Lomonosov Moscow State University and the computing complex of the resource sharing center of the National Research Center Kurchatov Institute.

1. Введение

Значительное количество механической энергии поступает в глубоководные области Мирового океана вследствие взаимодействия поверхностного прилива с топографией дна. В результате, в толще воды возникают т.н. инерционные и внутренние волны [1], которые вносят значимый вклад в перемешивание. Возникновение инерционных волн объясняется действием силы Кориолиса, связанной с вращением Земли. Внутренние волны в глубине океана возникают вследствие различных возмущений устойчивой плотностной стратификации. Внутренние волны обладают специфическим дисперсионным соотношением, в силу которого частота волны задает угол между волновым вектором и

направлением гравитационного поля. При отражении от границ жидкого объема волны меняют свое направление, сохраняя направление с вертикалью, а не нормалью к поверхности. Значительный интерес в этом контексте представляют волновые аттракторы, которые соответствуют случаям, при которых пучки внутренних волн притягиваются к замкнутой траектории. Принципиальная возможность возникновения подобных структур была впервые описана теоретически [2][3] и затем неоднократно проверена в ходе экспериментальных и численных исследований международными коллективами, в том числе из Голландии, Франции, США и России [3][8].

Простейший случай аттрактора внутренних волн в стратифицированной жидкости можно наблюдать в трапецеидальном контейнере (рис.1) при воздействии монохроматических возмущений. Подобная модельная конфигурация позволяет выделить ряд процессов, которые возникают в задачах океанологии из-за специфического закона отражения волновых пучков. В океане топография дна может быть представлена подводными хребтами, проливами, разломами, континентальными склонами либо фоновыми неровностями орографии дна океанских котловин. В рамках модельной системы возможно возникновение волновых аттракторов, схожих с таковыми в океане. Геометрическая структура аттракторов в трапециедальной области зависит от наклона волнового пучка (зависящего от соотношения между частотой возмущения и частотой плавучести), угла наклона одной из сторон и соотношения горизонтального и вертикального масштабов. Отметим, что аттрактор может существовать в классическом ламинарном режиме в окрестности лучевого «скелета», получающегося в результате решения задачи о биллиарде волновых лучей, лишь тогда, когда мощность внешнего воздействия достаточно мала. В том случае, когда мощность внешнего воздействия возрастает выше некоторого порога, течение достаточно быстро переходит к турбулентному режиму, и регулярные структуры течения размываются. Тем не менее, до сих пор остается открытым вопрос о том, для какого режима характерна наиболее высокая аккумуляция кинетической энергии волновых движений.

В работах [9][12] с помощью различных численных методов изучались волновые аттракторы для ситуации, когда длина области и ее высота являлись сопоставимыми величинами. В данных работах был получен ряд существенных с точки зрения гидродинамики выводов о механизме образования аттракторов и наличия стоящих и бегущих составляющих в волновых режимах при различных воздействиях. Представляет интерес вопрос о том, как происходит генерация волновых аттракторов в случае, когда горизонтальный масштаб аттрактора значительно превышает вертикальной. Такой случай ближе соответствует типичной геометрии Мирового океана.

В рамках настоящей работы мы исследуем результаты прямого численного моделирования, которые получены для большого соотношения между геометрическими параметрами исследуемой области. Для обработки результатов прямого численного моделирования применяются спектральные методы, связанные с оконным преобразованием Фурье и разложения по эмпирическим модам (Empirical Mode Decomposition).

2. Математическая постановка

Рассмотрим простейшую геометрическую конфигурацию в виде двумерной области, имеющей форму прямоугольной трапеции, в которой при определенном сочетании параметров могут возникать волновые аттракторы (рис. 1). Область имеет длину L, высоту H, геометрическая форма трапеции характеризуется безразмерным параметром $d = (L_1 - L_2)/L_1$, причем значения $d = 1$ и $d = -1$ отвечают предельным случаям прямоугольной и треугольной конфигураций, соответственно. Область заполнена стратифицированной жидкостью с постоянным вертикальным градиентом плотности. Плотность возрастает с глубиной, что согласуется с изменением солености в океане. Наклонная граница моделирует наклон рифа или материковый склон. Внешнее воздействие

задается в виде синусоидального возмущения верхней горизонтальной границы. Длина волны возмущения соответствует длине верхнего основания трапеции, амплитуда волны мала по сравнению с длиной.

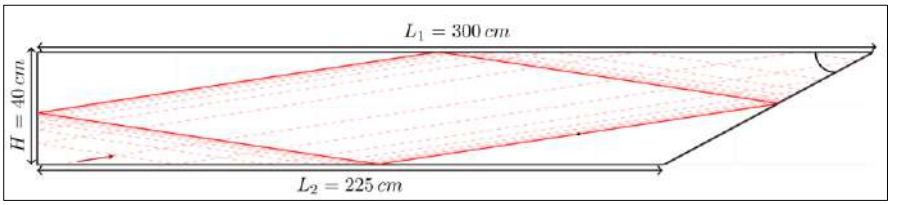

Рис. 1. Принципиальная схема сходимости пучков волн к волновому аттрактору и размеры области
Fig. 1. Scheme of convergence of internal wave beams to a wave attractor

Для решения задачи о волновых движениях в описанной области проведено прямое численное моделирование уравнений вязкой жидкости в поле силы тяжести. Уравнения сохранения импульса и диффузии соли в приближении Буссинеска [13]-[15] представляются в виде:

$$\frac{\partial \vec{V}}{\partial t} + (\vec{V}, \nabla)\vec{V} = \frac{-1}{\rho_m}\nabla \hat{p} + \nu \Delta \vec{V} + \vec{f},$$
$$\frac{\partial \rho_s}{\partial t} + \vec{V} \cdot \nabla \rho_s = \nabla \cdot \frac{\nu}{Sc}(\nabla \rho_s),$$
$$\nabla \cdot \vec{V} = 0.$$

Здесь $\vec{V}$ – вектор скорости с компонентами $\{v_x, v_y\}$; ν – кинематическая вязкость жидкости; ρ_m – значение плотности на верхней границе; ρ_s – добавка плотности, обусловленная наличием солености; приведенное давление $\hat{p}$ – разница между полным давлением и гидростатическим для пресной воды (при плотности ρ_m), $\vec{f} = (\rho_s/\rho_m)\vec{g}$, $Sc = \frac{\nu}{D}$ – число Шмидта, где D – коэффициент диффузии.

Верхняя граница совершает малые гармонические колебания в воде стоячей волны (задача при этом решается в фиксированной области):

$$h = a\sin(\omega_0 t)sin(2\pi x/L),$$

где a – его амплитуда, а ω_0 – циклическая частота. Указанное возмущение принято в качестве простейшей модели поверхностного баротропного прилива. В силу малости амплитуды колебаний внешнее воздействие можно представить в виде граничного условия на скорость:

$$v(H, y, t) = a\omega_0 \cos(2\pi x/L)\sin(\omega_0 t),$$

На всех границах (кроме верхней) выполняется условие прилипания для скорости $v_n = 0$, $v_\tau = 0$. Для солености на всех границах выполняется условие изоляции $\frac{\partial \rho}{\partial n} = 0$. Моделирование внутренних волн проводится при параметрах, позволяющих в дальнейшем провести экспериментальные исследования в лаборатории, поэтому постановка приводится в размерных переменных:

L = 300 мм,
H = 40 мм,
d = 0.5,
$\varepsilon = \frac{H}{L}$,
ω_0 = 0.0149 с$^{-1}$,
N =1 с$^{-1}$,

где N – частота плавучести (частота Брента-Вяйселя).

В работе под энергией системы мы будем понимать только кинетическую энергию. Наряду с упомянутыми характеристиками, мы будем пользоваться относительной частотой

$$\overline{\omega} = \frac{\omega_0}{N}$$

и относительной энергией

$$\overline{E} = \frac{E}{E_w},$$

где E – полная кинетическая энергия, рассчитываемая как интеграл по площади трапеции

$$E(t) = \iint \frac{\rho_m}{2}\left[v_x^2(t) + v_y^2(t)\right]dS,$$

а для нормировки используется величина

$$E_w = \rho_m S,$$

равная кинетической энергии жидкого объема, движущегося как единое целое со скоростью $a\omega$, (здесь S – площадь трапеции). Все графики приведены для случая $\varepsilon=0.13$, если не указано иное.

3. Анализ данных прямого численного моделирования

Прямое численное моделирование задачи производилось при помощи метода спектральных элементов на неравномерных сетках на основе модификаций открытого кода nek5000 [14].

Численное исследование показывает, что в зависимости от амплитуды внешнего воздействия возможны различные сценарии развития волновых режимов. Если амплитуда воздействий достаточно мала, то происходит формирование волнового аттрактора, и энергия аккумулируется вдоль замкнутой кривой. Аттрактор типа (1,1) существует лишь в ограниченном диапазоне частот, определяемом граничными режимами, в которых аттракторы вырождаются в диагонали трапеции [17].

При увеличении амплитуды начинается переход к слаботурбулентному режиму. В отличие от геометрии с сопоставимыми длинами в вертикальном и горизонтальном направлении, вместо дочерних волн, удовлетворяющих условию триадного взаимодействия [18], появляются существенные по величине супергармоники. Дальнейший рост амплитуды внешнего воздействия способствует выводу системы в область сильной волновой турбулентности.

Далее рассмотрим несколько характерных случаев в зависимости от амплитуды внешнего воздействия, демонстрирующих формирование различных режимов движения жидкости. Следует отметить, что динамика средней энергии в системе имеет вид высокочастотных осцилляций, наложенных на некоторую среднюю кривую. Вместе с тем для полноты картины хотелось бы иметь автоматизированный алгоритм, способный выделять последнюю. Это можно делать с помощью оконного преобразования Фурье [19][20] и разложения по эмпирическим модам, известного под названием Empirical mode decomposition (EMD) [16].

Оконное преобразование Фурье представляет собой наиболее простой способ анализа, но при уменьшении ширины окна и среднее значение, и гармоники перестают отражать физический смысл разложения Фурье. Таким образом, частотно-временная диаграмма является достаточно удобной для случая, если колебания наложены на относительно медленно меняющуюся функцию. Однако если истинное среднее испытывает значительные скачки производной, то построенное таким образом среднее будет вести себя достаточно нерегулярным образом. Опыт построения средней энергии таким способом в данной задаче оказался неудачным.

Более приемлемым оказывается разложение по эмпирическим модам, смысл которого состоит в разложении результата по набору функций (также известных под названием Intrinsic mode functions, или IMF), отвечающих следующим требованиям:

1) количество экстремумов и нулей используемых в разложении функции отличается не более чем на 1;

2) среднее значение, определяемое по верхней и нижней огибающей, должно равняться нулю.

Особенность процедуры построения эмпирических мод заключается в том, что первыми будут выявлены гармоники, отвечающие наиболее высоким частотам. Этот метод подходит для исследования широкого класса нелинейных сигналов, однако отдельные моды не имеют определенной физической интерпретации. Для выделения среднего из сигнала с неустойчивостью имеет смысл вычитать из него первую эмпирическую моду.

Численно вычисление мод происходит итерационным методом с помощью инструментов библиотеки PyEMD. Для интерполяции данных использовались кубические сплайны, однако использование других методов существенно не влияет на результат. Поскольку для построения эмпирической моды требуется верхняя и нижняя огибающие, строящиеся по локальным максимумам, то определение последним происходило с помощью параболической интерполяции. В качестве критерия остановки итерационного процесса выбиралось выполнение одного из условий: число итераций или малости среднеквадратичной разницы двух итераций, деленной на норму предыдущей.

В рамках настоящей работы частотные и частотно-временные диаграммы строятся с помощью оконного преобразования Фурье. Разумеется, подобный подход имеет те же недостатки, что и локальное осреднение. Тем не менее, эти диаграммы дают хорошее представление о качественном сценарии развития нелинейных процессов, поэтому для наших целей такой метод вполне подходит.

4. Результаты для различных амплитуд внешней силы

На рис. 2 представлены результаты расчетов при значении амплитуды a=0.01 мм, которое является достаточно малым, чтобы можно было говорить о ламинарном волновом режиме. На поле вертикальной скорости отчетливо виден волновой аттрактор, что говорит о ламинарном характере движений.

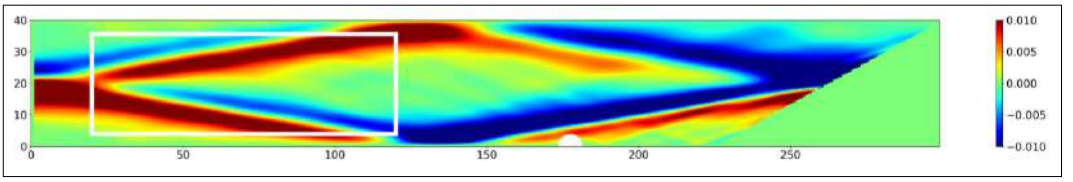

Рис. 2. Поле вертикальных скоростей при a=0.01 см. Белым обозначена область построения спектра
Fig. 2. Field of V_y at a=0.01 cm. White rectangular marks the area, where spectrum was calculated

Энергия в таком случае аккумулируется на аттракторе и после стремительного роста колеблется вокруг устойчивого среднего (показано черным, рис. 4a). Для сравнения приводим график зависимости энергии от времени для случая области, имеющей сопоставимые глубину и длину (ε=0.67), при наличии аттрактора того же вида: среднее значение безразмерной энергии оказывается на порядок ниже (рис. 4b).

Аттрактор может существовать лишь в ограниченном диапазоне частот [17], границы которого ограничены критическими частотами $\omega_{c1,2}$. На этих частотах форма аттрактора вырождается в диагональ трапеции. При этом аккумуляция энергии остается незначительной (рис. 3c-d). Кроме того, на рис. 3c-d хорошо видно, что кинетическая энергия системы колеблется между максимальным и минимальным значением, причем минимальное значение практически равно нулю. Известно, что для подобной системы волновой процесс имеет характер стоячих волн, что сильно отличает такую систему от случая прогрессивных волн, показанного на рис. 3a-b.

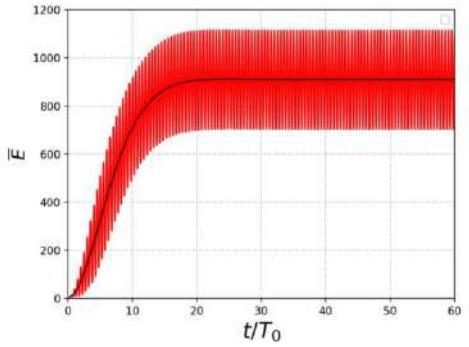

Рис. 3a. Относительная энергия при ε=0.13 и a=0.01 см
Fig. 3a. Relative energy at ε=0.13 and a=0.01 cm

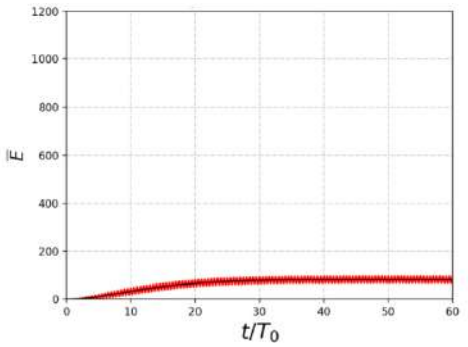

Рис. 3b. Относительная энергия для геометрии ε=0.67 при a=0.01 см
Fig. 3b. Relative energy for non-largeAR case at ε=0.67 a=0.01 cm

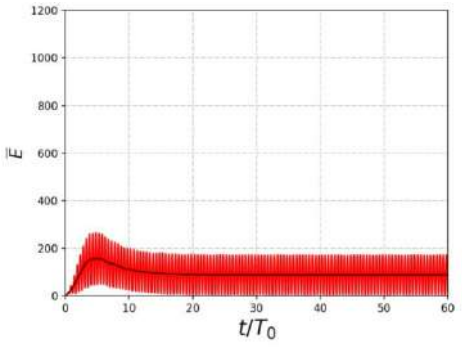

Рис. 3c. Относительная энергия на первой критической частоте ω_{c1}=0.13N при ε=0.13 и a=0.01 см
Fig. 3c. Relative energy on the second critical frequency ω_{c1}=0.13N at ε=0.13 and a=0.01

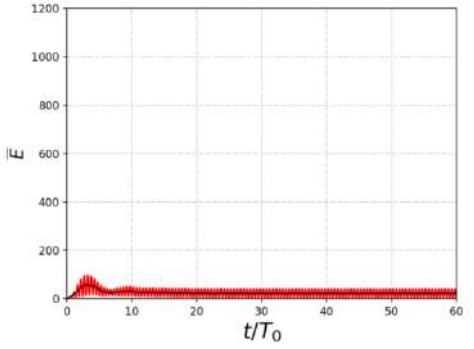

Рис. 3d. Относительная энергия на второй критической частоте ω_{c2}=0.18N при ε=0.13 и a=0.01см
Fig. 3d. Relative energy on the second critical frequency ω_{c2}=0.18N at ε=0.13 a=0.01

Тем не менее, даже в режиме с малым входным возмущением, при котором динамика системы близка к линейной (рис. 3а), можно наблюдать несколько пиков частотного спектра (рис. 4), который имеет линейчатый характер.

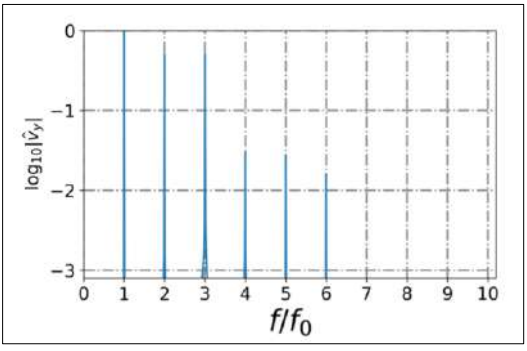

Рис. 4. Спектр при ε=0.13, a=0.01 см
Fig. 4. Spectrum at a=0.01 cm

Это, по-видимому, является следствием удлиненной формы области, поскольку в линейном случае при сравнимых масштабах высоты и горизонтальной длины (аспектное отношение

порядка единицы) частотный спектр волновго движения содержит лишь одну частоту, а при большой аплитуде возмущения бало зафиксировано наличие компоненты с удвоенной частотой [6]. Частотно-временной спектр в дотурбулентном режиме остается относительно простым, видны только составляющие на нескольких первых гармониках, кратных частоте возмущения (рис. 5).

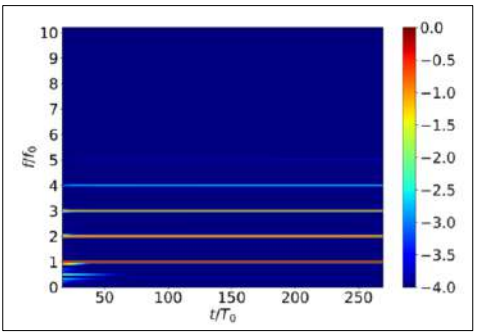

Рис. 5. Частотно-временная диаграмма
Fig. 5. Time-frequency diagram

Амплитуда a=0.14 мм соответствует переходному режиму между линейным ламинарным режимом и нелинейным режимом волновой турбулентности. При такой амплитуде волновой аттрактор визуально не просматривается (рис. 6).

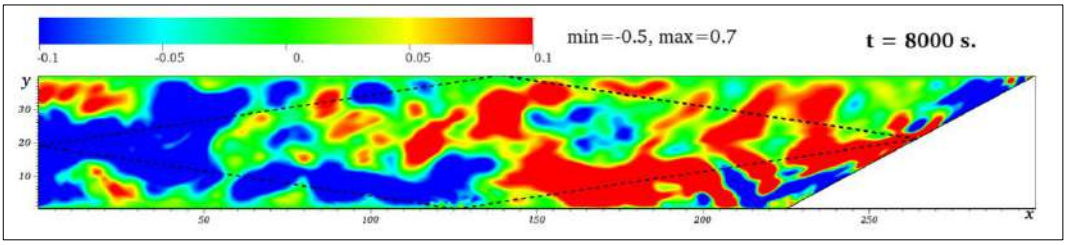

Рис. 6. Поле вертикальных скоростей при a=0.14 см
Fig. 6. V_y at a=0.14 cm

На рис. 6 видно, что показанная пунктиром структура аттрактора, характерная для линейного режима (рис. 2), полностью размыта, волновое движение имеет характер развитой волновой турбулентности. График зависимости безразмерной кинетической энергии $\overline{E}$ от времени, показан на рис. 7. На начальном этапе среднее значение энергии испытывает характерный рост, который останавливается на уровне $\overline{E}$~ 700, как и для слаботурбулентного режима, после чего величина энергии остается постоянной до значений времени порядка 125 периодов возмущающего воздействия. Интересный эффект наблюдается в процессе дальнейшей эволюции: по достижении (t ~ $125T_0$) среднее значение безразмерной энергии начинает с умеренной скоростью расти с одновременным увеличением амплитуды колебаний вокруг него. Можно предположить, что на больших временах происходит запуск каскада нелинейных волновых взаимодействий, вследствие чего в системе возникают существенные по величине субгармонические компоненты поля внутренних волн.

Вопрос о том, какому режиму движения соответствует наибольшее значение средней энергии, остается открытым, в расчетах, представленных на рис. 7, процесс на больших временах далек от насыщения, кроме того, нельзя исключать возможность появления следующего каскада нелинейных волновых взаимодействий на больших временных масштабах. На первый взгляд кажется, что развитие волновой турбулентности в системе вследствие каскада волновых взаимодействий должно «размывать» все структуры и мешать повышению среднего энергии в системе, поскольку дочерние волны подпитываются

энергией из основной волны, соответствующей частоте возмущающего воздействия. Однако представленные на рис. 7 результаты расчетов показывают, что в режиме слабой волновой турбулентности [21] возможна «накачка» системы с увеличением среднего уровня энергии.

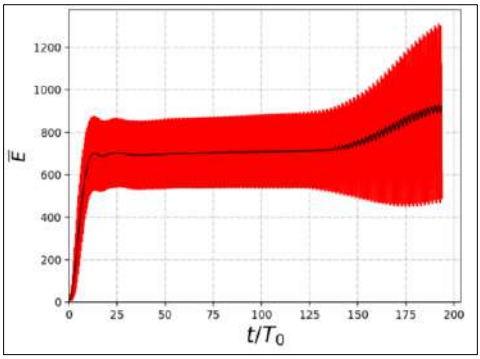

Рис. 7. Относительная кинетическая энергия при a=0.14 см
Fig. 7. Relative kinetic energy at a=0.14 cm

Режим развитой волновой турбулентности в изучаемой системе соответствует значению a=0.20 мм. Для наглядности на графике энергии (рис. 8) приведены первые две эмпирические моды. В сценарии развития турбулентного режима можно выделить следующие этапы: 1) первичная аккумуляция волновой энергии к волновому аттрактору. При этом неустойчивость ламинарного волнового аттрактора начинает развиваться еще до достижения максимума средней кинетической энергии; 2) установление первичного турбулентного режима, при котором вначале средняя кинетическая энергия и величина пульсаций несколько проседают, но затем на протяжении примерно 50-ти периодов внешних воздействий в среднем качественно не меняются; 3) на фоне сформировавшегося во втором этапе режима начинается рост новой неустойчивости. На рис. 8 он проявляется в виде роста второй моды EMD разложения. Удивительно, что при этом начинает существенно расти не только величина пульсаций, но и сама средняя кинетическая энергия.

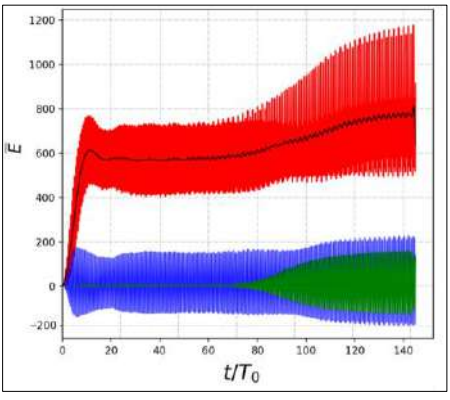

Рис. 8. Кинетическая энергия при a=0.20 см
Fig. 8. Kinetic energy at a=0.20 cm

На рис. 9a-c отражена зависимость осредненных по жидкому объему интегральных величин от амплитуды внешнего воздействия. Полная кинетическая энергия (рис. 9a) и диссипация (рис. 9b) растут с увеличением амплитуды возмущающего воздействия. Относительная энергия $\overline{E}$ демонстрирует необычное поведение. Немонотонное поведение зависимости относительной энергии от амплитуды, вероятно, связано со сменой характера волновой турбулентности.

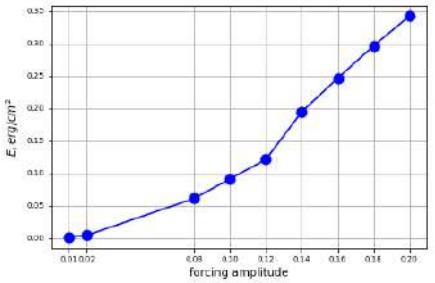

Рис. 9a. Средняя кинетическая энергия (полная энергия, нормированная на объем области и осредненная по времени после установления периодического режима)
Fig. 9a. Mean energy

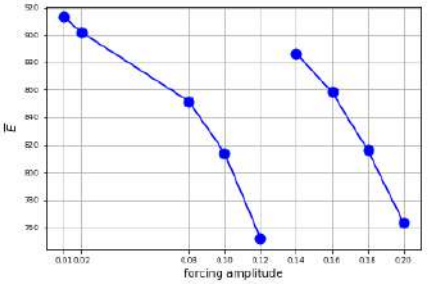

Рис. 9b. Относительная энергия
Fig. 9b. Relative energy

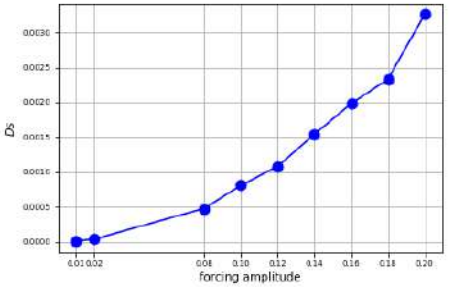

Рис. 9c. Диссипация
Fig. 9c. Dissipation

4. Заключение

Возникновение волновых аттракторов при большом соотношении между длиной и глубиной стратифицированного водоема (расчетной либо экспериментальной области) представляет собой большой интерес как с практической, так и с теоретической точек зрения. По сравнению с изученным на лабораторных масштабах модельным случаем областей, имеющих сопоставимые масштабы в вертикальном и горизонтальном направлениях, случай большого аспектного соотношения соответствует более типичной для природных систем (океан, озеро) геометрии. В случае удлиненной в горизонтальном направлении геометрии оказывается, что величина безразмерной энергии, аккумулированной в системе, может возрастать еще на порядок. При этом даже при сравнительно малой амплитуде входного воздействия имеется тенденция к развитию в системе неустойчивости, при которой возникает частотные спектры, имеющие развитый набор супергармоник.

В режимах развитой волновой турбулентности в расчетах наблюдается неожиданно высокое значение средней безразмерной энергии системы, т.е. имеет место «накачка» системы до высокого энергетического уровня, причем имеет место полная потеря визуальной структуры волновых аттракторов, а также уменьшение относительного вклада компоненты, соответствующей частоте возмущения, в суммарную энергию системы. Проведенные исследования наглядно показали, что несмотря на то, что в турбулентных режимах с большим отношением масштабов визуально аттрактор не наблюдается, концентрация кинетической энергии оказывается даже выше, чем в ламинарных режимах.

Список литературы / References

[1]. Garrett C.J.R., Munk W. H. Internal waves in the ocean. Annual Review of Fluid Mechanics, vol. 11. 1979, pp. 339-369.

[2]. Maas L.R. M., Lam F.P.A. Geometric focusing of internal waves. Journal of Fluid Mechanics, vol. 300, 1995, pp. 1-41.

[3]. Maas L. R.M., Benielli D., Sommeria J., Lam F.P.A. Tidally generated internal-wave attractors between double ridges. Nature, vol. 388, 1997, pp. 557-561.

[4]. Scolan H., Ermanyuk E., Dauxois T. Nonlinear Fate of Internal Wave Attractors. Physical Review Letters, vol. 110, 2013, article id 234501.

[5]. Maas L.R.M. Wave focusing and ensuing mean flow due to symmetry breaking in rotating fluids. Journal of Fluid Mechanics, vol. 437, 2001, pp. 13-28.

[6]. Grisouard N., Staquet C., Pairaud I. Numerical simulation of a two-dimensional internal wave attractor. Journal of Fluid Mechanics, vol. 614, 2008, pp. 1-14.

[7]. Hazewinkel J., Grisouard N., Dalziel S. B. Comparison of laboratory and numerically observed scalar fields of an internal wave attractor. European Journal of Mechanics – B/Fluids, vol. 30, issue 1, 2011, pp. 51-56.

[8]. Брузе К., Доксуа Т., Ерманюк Е. и др. Прямое численное моделирование аттракторов внутренних волн стратифицированной жидкости в трапецеидальной области с колеблющейся вертикальной стенкой. Труды ИСП РАН, том 25, вып. 5, 2014 г., стр. 117-142 / Brouzet C., Dauxois T., Ermanyuk E. et al. Direct numerical simulation of internal gravity wave attractor in trapezoidal domain with oscillating vertical wall. Trudy ISP RAN/Proc. ISP RAS, vol. 25, issue 5, 2014, pp. 117-142 (in Russian). DOI: 10.15514/ISPRAS-2014-26(5)-6.

[9]. Beckebanze F., Brouzet C., Sibgatullin I.N., Maas L.R.M. Damping of quasi-two-dimensional internal wave attractors by rigid-wall friction. Journal of Fluid Mechanics, vol. 841, 2018, pp. 614-635.

[10]. Сибгатуллин И.Н., Ерманюк Е.В., Ватутин К.А., Рязанов Д.А., Сюй С. Численное моделирование трехмерных волновых аттракторов. Океанологические исследования, vol. 47, № 1, 2019 г., стр. 112-115 / Sibgatullin I.N., Ermanyuk E.V., Vatutin K.A., Ryazanov D.A., Xu X. Numerical simulation of three-dimensional wave attractors. Journal of Oceanological Research, vol. 47, № 1, 2019, pp. 112-115 (in Russian).

[11]. Провидухина М., Сибгатуллин И. Применение спектральных методов обработки данных к результатам численного моделирования аттракторов внутренних волн. Труды ИСП РАН, том 28, вып. 1, 2016 г., стр. 275-282 / Providukhina M., Sibgatullin I. Application of statisticaland spectral methods to computational modeling of internal wave attractors. Trudy ISP RAN/Proc. ISP RAS, 2016, vol. 28, issue 1, pp. 275-282 (in Russian). DOI: 10.15514/ISPRAS-2016-28(1)-16.

[12]. Сибгатуллин И.Н., Ерманюк Е.В. Аттракторы внутренних и инерционных волн (обзор). Прикладная механика и техническая физика, № 2, 2019 г., стр. 113-136 / Sibgatullin I.N., Ermanyuk E.V. Internal and inertial wave attractors: a review. Journal of Applied Mechanics and Technical Physics, № 2, 2019, pp. 113-136 (in Russian).

[13]. Kuznetsova D. V., Sibgatullin I. N. Transitional regimes of penetrative convection in a plane layer. Fluid Dynamics Research, vol. 44, no. 3, 2012, article id 031410.

[14]. Fischer P., Ronquist E. Spectral element methods for large scale parallel Navier-Stokes calculations, Computer Methods. Applied Mechanics and Engineering, vol. 116, issue 1-4, 1994, pp. 69-76.

[15]. Spiegel E.A., Veronis G. On the Boussinesq Approximation for a Compressible Fluid. Astrophysical Journal, vol. 131, p. 442.

[16]. Huang N.E. et al. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proceedings of The Royal Society A. Mathematical Physical and Engineering Sciences, vol. 454, 1998, pp. 903-995.

[17]. Рязанов Д.А., Провидухина М.И., Сибгатуллин И.Н., Ерманюк Е. В. Бигармонические аттракторы внутренних гравитационных волн. Механика жидкости и газа, принята в печать, 2020.

[18]. Dauxois T., Joubaud S., Odier P., Vanaille A. Instabilities of internal gravity wave beams. Annual Review of Fluid Mechanics, vol. 50, 2018, pp. 131–156.

[19]. Allen J.B. Short Time Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 25, issue 3, 1977, pp. 235–238.

[20]. Jacobsen E., Lyons R. The sliding DFT. Signal Processing Magazine, vol. 20, issue 2, 2003, pp. 74–80

[21]. Nazarenko S. Wave turbulence. Springer, 2011, 295 p.

[22]. Koudella C. R., Staquet C. Instability mechanisms of a two-dimensional progressive internal gravity wave. Journal of Fluid Mechanics, vol. 548, 2006, pp. 165–196.
[23]. Sutherland B. R. Internal wave instability: wave-wave and wave-induced mean flow interactions. Physics of Fluids, vol. 18, issue 7, 2006, article id 074107.
[24]. Bourget B., Dauxois T., Joubaud S., Odier P. Experimental study of parametric subharmonic instability for internal plane waves. Journal of Fluid Mechanics, vol. 723, 2013, pp. 1–20.
[25]. Bourget B., Scolan H., Dauxois T. et al. Finite-size effects in parametric subharmonic instability. Journal of Fluid Mechanics, vol. 759, 2014, pp. 739–750.
[26]. Karimi H.H., Akylas T.R. Parametric subharmonic instability of internal waves: locally confined beams versus monochromatic wave trains. Journal of Fluid Mechanics, vol. 757, 2014, pp. 381–402.
[27]. Brouzet C., Ermanyuk E.V., Joubaud S. et al. Internal wave attractors: different scenarios of instability. Journal of Fluid Mechanics, vol. 811, 2017, pp. 544–568.
[28]. Dauxois T., Ermanyuk E. V., Brouzet C. et al. Abyssal mixing in the laboratory. In The ocean in motion: circulation, waves, polar oceanography, Springer, 2018, pp. 221–237.

Информация об авторах / Information about authors

Степан Алексеевич ЕЛИСТРАТОВ – студент, его научные интересы включают особенности волновых течений из-за периодических воздействий на стратифицированные жидкости и приложения к океанологии и астрофизики.

Stepan Alekseevich ELISTRATOV is a student, his scientific interests include wave motions due to periodic external forcing in astrophysical and oceanological applications.

Кирилл Александрович ВАТУТИН – аспирант, его научные интересы включают задачи волновой динамики в стратифицированных и вращающихся средах, разработка программных средств для задач гидроаэродинамики.

Kirill Alexandrovich VATUTIN is a PhD student, his scientific interests include the problems of wave dynamics in stratified or rotating media, and development of CFD software.

Ильяс Наилевич СИБГАТУЛЛИН – старший научный сотрудник, кандидат физико-математических наук. Сфера научных интересов: гидродинамика конвективных и турбулентных течений, нелинейная динамика, внутренние и инерционные волны, волновые аттракторы.

Ilias Nailevich SIBGATULLIN – is a senior researcher, PhD. Research interests: convection, turbulence, nonlinear dynamics, internal and inertial waves, wave attractors, wave turbulence.

Евгений Валерьевич ЕРМАНЮК – директор института, доктор физико-математических наук. Научные интересы включают, в том числе: гидродинамический эксперимент, стратифицированные жидкости, внутренние волны, плотностные течения, гидродинамический удар, оптические измерения в жидкостях.

Evgeniy Valerievich ERMANYUK – director of the institute, doctor, professor. Research interests: hydrodynamic experiments, stratified fluids, internal and inertial waves.

Евгений Александрович МИХАЙЛОВ является ассистентом, кандидатом физико-математических наук, его научные интересы включают изучение двумерных моделей в гидродинамике с помощью численных и аналитических подходов, исследование астрофизических моделей.

Evgeny Aleksandrovich MIKHAILOV is an assistant professor, Ph.D. His research interests include two-dimensional models in hydrodynamics using numerical and analytical methods, elaboration of astrophysical models.