Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

МОЛОДАЯ ИНФОРМАТИКА

Вып. 3

СБОРНИК ТРУДОВ
АСПИРАНТОВ И МОЛОДЫХ УЧЕНЫХ

Под редакцией
к.ф.-м.н. А.Ю. Пальянова

Новосибирск 2011

Сборник содержит статьи, представленные аспирантами и молодыми сотрудниками ИСИ СО РАН, по следующим направлениям: теоретические аспекты программирования, информационные технологии и информационные системы, системное программное обеспечение, прикладное программное обеспечение.

Siberian Branch of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

YOUNG INFORMATICS

Issue 3

COLLECTION OF PAPERS
OF GRADUATE STUDENTS
AND YOUNG SCIENTISTS

Edited by
Palyanov A.Yu., Ph.D.

Novosibirsk 2011

The volume contains the papers presented by post-graduates and young researchers of A.P. Ershov Institute of Informatics Systems which concern the following research areas: theoretical aspects of programming, information technologies and systems, system software and application software.

ПРЕДИСЛОВИЕ

Цель сборника – стимулирование научной деятельности аспирантов и молодых сотрудников (до 35 лет) Института систем информатики СО РАН и их обучение качественному представлению научных работ. При обучении использовалось двухэтапное рецензирование работ, и в сборник включались те статьи, которые были доработаны с учетом рецензий. Работы принимались в рамках тематики института по следующим направлениям: теоретические аспекты программирования, информационные технологии и информационные системы, системное программное обеспечение, прикладное программное обеспечение.

Целью работы «**Трассовые эквивалентности временных сетей Петри**» является анализ поведения параллельных систем реального времени – сложная задача, решение которой требует привлечения формальных методов и средств. За последние десятилетия с этой целью были разработаны различные модели, учитывающие временные характеристики функционирования систем: временные автоматы, временные сети Петри, временные структуры событий и т.д. Понятие времени также было введено и в поведенческие эквивалентности. В данной работе определяется и исследуется семейство трассовых эквивалентностей в семантиках «интерливинг/частичный порядок» в контексте временных сетей Петри. Изучаемые эквивалентности основываются на понятии временного процесса, т.е. временного расширения причинной сети за счет сопоставления глобальных моментов срабатывания переходов. При этом рассматриваются только корректные по времени процессы, т.е. такие, временная функция которых удовлетворяет специально разработанным свойствам корректности. Устанавливаются взаимосвязи эквивалентностей и строится иерархия классов эквивалентных временных сетей Петри.

Работа «**Моделирование биологических нейронных сетей с использованием NeuroML**» выполнялась в рамках сотрудничества с международным проектом OpenWorm по созданию первого виртуального организма, нематоды C. elegans, на основе детальных биологических экспериментальных данных о его строении, включая нервную, мышечную и сенсорную системы. Задача, о которой идет речь в данной работе – разработка алгоритмов для обработки промежуточных данных и создание с их помощью блока данных в формате NeuroML (один из сложившихся стандартов детального описания биологических нейронов и сетей нейронов, основанный на XML) для дальнейшего использования как в проекте OpenWorm, так и

для предоставления возможности использования этих материалов для других задач, связанных с исследованиями/моделированием этого организма.

Целью работы «**О методе выявления синонимичных конструкций естественного языка и его применении к задаче информационного поиска**» является разработка метода, позволяющего сопоставлять конструкции естественного языка и отождествлять перефразированные варианты предложений, основываясь на анализе их синтаксической структуры. Метод разрабатывается в предположении, что на вход подаются синтаксические диаграммы двух сравниваемых предложений, и ориентирован на использование системы синтаксического анализа предложений на английском языке Link Grammar Parser (Temperley et. al., 1998). Ее особенностью является то, что получив предложение, она приписывает ему синтаксическую структуру, которая состоит из множества помеченных связей, соединяющих пары слов. Пометка каждой связи соответствует некоторому случаю правильного употребления данной пары слов в предложении и одновременно некоторому семантико-синтаксическому отношению между элементами предложения.

В работе «**О реализации алгоритма иерархического кластерного анализа на GPU средствами технологии CUDA**» рассматривается алгоритм иерархической кластеризации и предлагается метод отображения данного алгоритма на параллельную мультипроцессорную систему, использующуюся на современных графических процессорах GPU. Для реализации алгоритма используется технология параллельных вычислений CUDA, разработанная компанией NVIDIA, предоставившая доступную возможность задействовать вычислительные мощности графических процессоров для решения сложных расчетных задач. В работе делаются оценки времени выполнения алгоритма иерархической кластеризации в последовательном случае, параллельном случае для абстрактной параллельной машины и на GPU. Также получены соответствующие коэффициенты ускорения. Процедура построения матрицы расстояний между кластерами, реализованная на GPU средствами системы CUDA, при больших размерностях задачи более чем в 60 раз выигрывает по производительности в сравнении с той же программой на C++ в случае ее реализации на центральном процессоре.

Работа «**Обзор методов представления семантики текстов и извлечения знаний из них**» посвящена обзору ряда методов формального представления знаний и семантики текстов в целом, а также способов извлечения знаний из неструктурированных текстов. Про «сильные» методы представления семантики известно, что из-за большой сложности и трудоемкости реализации таких подходов ни одна из попыток их применения по сей

день не увенчалась успехом. Основной объект рассмотрения данной работы – значительно более реализуемые на практике «слабые» методы представления семантики: фреймовая модель в RCO Fact Extractor, фреймовая модель Симакова представления знаний, семантические сети в WOCADI Parser, реляционно-ситуационная модель текста в ИПС Exactus, лексико-синтаксические шаблоны в поисковой системе SEUS, подход к извлечению фактов из текста на основе онтологии.

В работе «**Стрип-преобразование сигналов и некоторые вычислительные эксперименты**» исследовался метод повышения помехоустойчивости, известный под названием «стрип-метод» (Мироновский, Слаев, 2006). Его суть заключается в предварительном преобразовании сигнала на передающем конце путем «разрезания» его на участки равной длительности, формирования их линейных комбинаций и обратного «склеивания» в единый сигнал той же или большей длительности. Интересно, что стрип-преобразование обладает свойством, напоминающим голографический эффект: после преобразования часть сигнала может быть полностью потеряна или уничтожена, однако при обратном преобразовании сигнал восстанавливается полностью, но с некоторыми отклонениями значений от исходных. В стрип-методе обычно используются матрицы Адамара (множество видов), а также можно использовать матрицы других видов. В настоящее время не ясно, какие матрицы лучше подходят для тех или иных типов сигналов. Для получения новых знаний в этой области в статье рассмотрены различные варианты стрип-преобразования сигналов на основе некоторых матриц Адамара и Хаара, широко применяемых в области обработки сигналов, изображений и в оптике. Произведены расчеты для конкретных типов простых сигналов, определены погрешности, возникающие после восстановления сигнала.

В работе «**Использование CQRS-технологии при разработке корпоративных приложений**» рассматривается вопрос перспективности применения CQRS-метода при разработке произвольных стандартных корпоративных приложений. Метод основан на применении архитектурного шаблона Command and Query Responsibility Segregation (CQRS) и сводится к тому, что все коммуникации осуществляются через систему сообщений, заранее определенной структуры, и в результате весь процесс взаимодействия клиентской и серверной частей приложения укладывается в две цепочки событий. Это позволяет избегать лишних преобразований и снижать сетевой трафик, что является весьма актуальным для разнообразных мобильных устройств (таких, как коммуникаторы и планшетные компьютеры, использующие различные операционные системы), в свете того, что поль-

зовательский рынок программного обеспечения демонстрирует спрос на приложения для корпоративных систем (складские, бухгалтерские, производственные и прочие отраслевые программы), пригодные к использованию на этих устройствах.

В работе **«Экспертная система определения заболевания по хроматограмме образца сыворотки крови»** рассматривается высокоэффективная жидкостная хроматография (ВЭЖХ) с многоканальным детектированием и, в частности, с многоволновой фотометрией в УФ области спектра, как современный эффективный метод обзорного анализа сыворотки крови человека. После обработки хроматографической и спектральной информации становится возможным отслеживать значимые изменения в составе крови и тем самым проводить диагностику ряда заболеваний, включая онкологические. В данной работе предложен вариант компьютерной системы, способной обрабатывать результаты такого анализа. Она включает этапы предварительной обработки сигнала с целью фильтрации шумов, разбиение хроматограммы на пики, сравнение полученных наборов пиков, а также ряд вспомогательных функций. Текущим результатом работы стала готовая легко расширяемая программная платформа, обеспечивающая удобный доступ к хроматографическим данным, их обработку базовыми алгоритмами и визуализацию результатов анализа. Алгоритмы, реализованные на данный момент, позволяют проводить фильтрацию входных данных, кластеризацию (как в автоматическом режиме, так и вручную), идентификацию веществ по базе спектральных данных, выполнять процедуру валидации хроматографа и обрабатывать хроматограммы в пакетном режиме.

PREFACE

The aim of this compilation is to encourage the research activities of post-graduates and young fellow researchers of the SB RAS Institute of Informatics Systems (up to the age of 35) and give them a chance to practice presenting their works in a quality manner. The training procedure involved two-step peer reviewing; the compilation contains only those works which have been improved according to the reviews. The topics of the accepted works pertain to the research areas of the Institute, including the following: theoretical aspects of programming: information technologies and information systems; systems software; applied software.

The work **"Trace equivalences of temporal Petri nets"** is an attempt to analyze the behavior of real time parallel systems, a complex task requiring the use of formal methods and tools. In recent decades several models have been developed for the purpose, taking into account the temporal characteristics of the systems' functioning – temporal automata, timed Petri nets, temporal event structures, etc. The notion of time has been implemented in behavioral equivalencies. The authors define and study a family of track equivalencies in interleaving/partial order semantics in the context of temporal Petri nets. The equivalencies under study are based on the concept of temporal process, i.e. temporal extension of the net in question by collating global moments of transition firing. Here only temporally correct processes are examined, i.e. such that their temporal function complies with the specially developed correctness properties. The interconnections of equivalencies are defined, and a class hierarchy of equivalent timed Petri nets is built.

The work **"Simulation of biological neural networks with NeuroML technology"** was written as part of collaboration within the international OpenWorm project aimed at the creation of the first virtual organism, the *Caenorhabditis elegans* nematode, based on detailed experimental data on its biology and physiology, including its nervous, locomotory, and sensory systems. The task discussed in this work is the development of algorithms for processing intermediate data and application of these algorithms to create a block of data in NeuroML format (one of the established XML-based standards for detailed description of biological neurons and neuron networks) for further use in OpenWorm, also making them available for use in other tasks related to the study and modeling of *C. elegans*.

The authors of **"On method of synonymous constructions of natural language revelation and its application to the problem of information retrieval"** aimed at developing a method of comparing natural language constructs and matching paraphrased variants of sentences based on the analysis of their syntactic structure. The method is developed based on the assumption that the input receives the diagrams of the two sentences being compared; it is oriented towards the use of the Link Grammar Parser sentence syntax analysis system (Temperley et al., 1998). A peculiar feature of the system is that once it has received a sentence, it ascribes it a corresponding syntactic structure consisting of a number of marked links connecting pairs of words. Each link park corresponds to a particular case of correct usage of the given word pair in a sentence and at the same time to a certain semantic-syntactic relationship between the sentence elements.

In "**On realization of hierarchical cluster analysis algorithm on GPU using CUDA technology**" the authors explore an algorithm of hierarchical clustering and propose a method of reflecting the algorithm onto a parallel multiprocessor system used on modern graphic processors (GPU). To implement the algorithms the authors use the CUDA parallel computation technology developed by NVIDIA; the technology enables the use of graphic processor computation power to perform complex computational tasks. The work provides estimates for the execution of hierarchical clustering algorithm in the sequential and parallel cases, using an abstract parallel machine and GPU. The corresponding acceleration rates have been obtained. Computational performance of constructing a matrix of distances between clusters implemented on GPU using the CUDA system is over 60 times higher than the same C++ program on CPU."

The work **"Review of methods of text semantics representation and extraction of knowledge from them"** is a review of a number of methods of formal knowledge and general text representation, as well as methods of mining data from unstructured texts. "Strong" methods of semantics representation are known to be extremely complex and demanding, with no success in practical application so far. The main objects of the paper are the following: frame model in RCO Fact Extractor, Simakov data representation frame model, WOCADI Parser semantic webs, relation-situational text model in Exactus, lexical and syntactical templates in SEUS search system, and ontology-based fact extraction approach.

The authors of "**Stripe-transformation of signals and some computational experiments**" have studied a method of interference immunity known as the strip method (Mironovskiy, Slaev, 2006). The essence of this method is to transform the signal on the transmitting end by "cutting" it into parts of equal lengths, forming their linear combinations and reverse "glueing" them into a single signal of the same or larger length. It's noteworthy that the strip transformation has a quality reminiscent of the holographic effect: upon the transformation a part of the signal can be completely lost or destroyed, yet the reverse transformation recovers the signal completely with some deviations from the original. The strip method employs Adamar matrices (a number of types); other types of matrices can be used as well. It's not yet clear which matrices are better suited for specific signal types. To obtain new knowledge in this area the authors review different types of strip transformation of signals based on Haar and Adamar matrices widely used in signal and image processing as well as in optics. Calculations have been done for several specific types of simple signals in order to determine error margins after signal extraction.

The work **"Employment of CQRS technology in development of corporate applications"** discusses the issue of exploitability of the CQRS method in developing random standard corporate applications. The method is based on the application of the Command and Query Responsibility Segregation (CQRS) architectural template; essentially in consists in having all communications done through a system of messages of a predefined structure; as the result, the whole client-server interaction within the application is reduced to two chains of events. This allows to avoid unnecessary transformations and reduce network traffic, which is a very relevant issue for mobile devices (sich as communicators and tablet PCs using different operating systems) since the software market is showing an increased demand for corporate applications compatible with such devices.

The work **"Expert system determining diseases by means of analysis of blood serum chromatogram"** explores a highly efficient method of liquid chromatography with multi-channel detection and multiwave UV photometry as a modern and efficient tool of analysis of synoptic human serum panels. Data from the processing of chromatographic and spectral information enables tracking significant changes in blood parameters and thus diagnosing a number of condition, including oncologic diseases. The authors propose a variant of a computer system capable of processing the results of such analysis. It consists of sev-

eral stages including preliminary signal processing to filter out noises, splitting the chromatogram into peaks, comparing sets of peaks as well as a number of auxiliary functions. The current result is a functional, extendable program platform enabling easy access to chromatographic data as well as the processing of the data and visualizing the analysis results. The currently implemented algorithms allow to filter input data, clustering (both automatic and manual) and substance identification using a spectral database, to validate the chromatograph device and to process batches of chromatograms.

Е.Г. Барам

ЭКСПЕРТНАЯ СИСТЕМА ОПРЕДЕЛЕНИЯ ЗАБОЛЕВАНИЯ ПО ХРОМАТОГРАММЕ ОБРАЗЦА СЫВОРОТКИ КРОВИ

1. ВВЕДЕНИЕ

Достижения в области информационных технологий, наблюдаемые в последние 20 лет, способствовали прогрессу во многих областях науки, в частности, в аналитической химии, и особенно в тех ее разделах, которые связаны с обработкой большого объема экспериментальных данных. Одним из таких разделов является жидкостная хроматография – метод разделения веществ в растворе, который впервые ввел в практику М.С. Цвет в 1903 году. Суть метода заключается в следующем: в верхнюю часть *хроматографической колонки,* представляющей из себя трубку, наполненную мелкодисперсным адсорбентом, помещают небольшую порцию раствора анализируемого образца и промывают колонку подходящим растворителем. Важно, чтобы молекулы компонентов образца в данном растворителе быстро адсорбировались и десорбировались с поверхности сорбента. В этом случае молекулы каждого типа будут передвигаться по колонке в виде узких концентрационных зон со скоростью, обратно пропорциональной силе адсорбции. Очевидно, что, если сила взаимодействия адсорбата с адсорбентом для молекул разных веществ будет различной, то и скорости движения зон этих веществ будут различаться, т.е. вещества, проходя через колонку, будут разделяться.

Скорость движения зоны вещества зависит от скорости движения растворителя (подвижной фазы, элюента), от химического строения адсорбента и вещества, от состава элюента, от температуры.

Зависимость величины концентрации вещества вдоль зоны в идеальном случае описывается уравнением Гаусса:

$$C(V) = h \cdot e^{-0.5\left(\frac{V-V_R}{\sigma}\right)^2}.$$

Наблюдаемый *хроматографический пик*, соответствующий *хроматографической зоне,* изображен на рис. 1. Если измерять концентрацию веществ в растворе на выходе колонки, то мы получим кривую, которая называется *хроматограммой*.

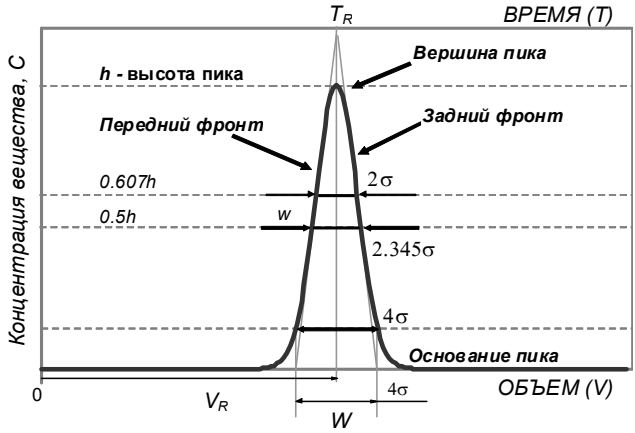


Рис. 1. Хроматографический пик
σ – стандартное отклонение, ***h* –** высота гауссова пика

Таким образом, хроматограмма является функцией зависимости концентрации вещества в растворе от объема, пропущенного через колонку растворителя или от времени. Каждому веществу на хроматограмме соответствует свой пик. Начало координат соответствует моменту ввода пробы (образца) в колонку. Абсцисса вершины пика называется *объемом удерживания вещества* (V_R), величина которого определяется химическим строением этого вещества, составом *подвижной фазы,* свойствами адсорбента (*неподвижной фазы*) и температурой. Когда говорят о *времени удерживания* (T_R), то имеют в виду, что $T_R = V_R / F$ где F – скорость потока растворителя через колонку. Мерой количества вещества, введенного в колонку, является площадь хроматографического пика, равная

$$S = \int_{V_1}^{V_2} C(V)dV ,$$

где V_1 и V_2 – «начало» и «конец» хроматографического пика.

Концентрация вещества в подвижной фазе, вытекающей из колонки (*элюат*), измеряется с помощью *детектора*, представляющего собой специальное устройство с проточной измерительной ячейкой, выходной сигнал которого пропорционален концентрации вещества в растворе. Устройство детектора может быть основано на многих физико-химических принципах, но мы рассмотрим только *фотометрический детектор*, который

применяется в хроматографе «Милихром А-02» (ЗАО «ЭкоНова», г. Новосибирск); такой детектор использовался в данной работе для разделения смесей веществ.

Регистрация оптической плотности при одной длине волны является простейшим способом детектирования, при котором можно лишь определить количество вещества по площади пика, но нельзя идентифицировать это вещество по его спектральным характеристикам. Детектор хроматографа «Милихром А-02» может работать в многоволновом режиме, быстро перестраивая монохроматор по циклической программе (до восьми длин волн в цикле) так, что концентрация вещества в элюате, протекающем через фотометрическую ячейку, успевает измениться лишь незначительно.

В настоящее время среди многих методов химического анализа заметное место занимает высокоэффективная жидкостная хроматография (ВЭЖХ) с многоканальным детектированием и, в частности, со многоволновой фотометрией в УФ области спектра. Интересным и важным направлением ВЭЖХ-УФ в последние годы можно считать «стандартизацию» обзорного анализа сыворотки крови человека. После обработки хроматографической и спектральной информации становится возможным отслеживать значимые изменения в составе крови и, тем самым, проводить диагностику ряда заболеваний, включая онкологические.

В данной работе мы предложим вариант компьютерной системы, способной обрабатывать результаты такого анализа. В этом процессе выделяются два основных этапа: обработка сигнала детектора – сглаживание, приведение к единой временной сетке, удаление выбросов и кластеризация и последующий анализ полученных компонентов с целью сравнения их с эталоном или идентификации по базе данных «ВЭЖХ-УФ». Идентификация компонентов может производиться по времени удерживания и спектральным отношениям, что дает возможность различить более чем 10^{11} веществ. Общая схема работы системы представлена на рис. 2:

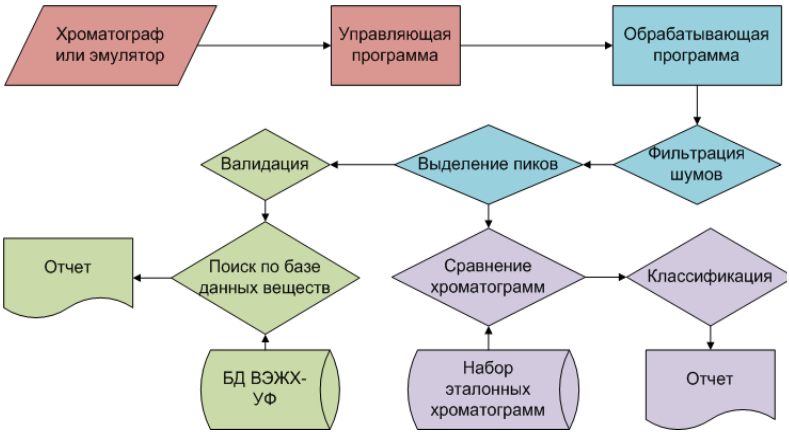


Рис. 2. Общая схема системы

2. МЕТОДЫ

На рис. 3 приведен пример данных, получаемых с хроматографа в 8-волновом режиме. Массив данных содержит в общей сложности около 22 000 точек.

Как видно из рисунка, сигнал детектора имеет относительно высокий уровень шума, что, разумеется, затрудняет обнаружение пиков. Кроме того, одной из главных проблем при проведении такого рода анализов является то, что каждый пик может иметь достаточно существенный дрейф (до 10% по времени и до 0,4° по спектральному углу) и накладываться на соседние пики.

Разрабатываемая нами система учитывает эти особенности и позволяет добиваться приемлемых результатов на широком спектре входных данных.

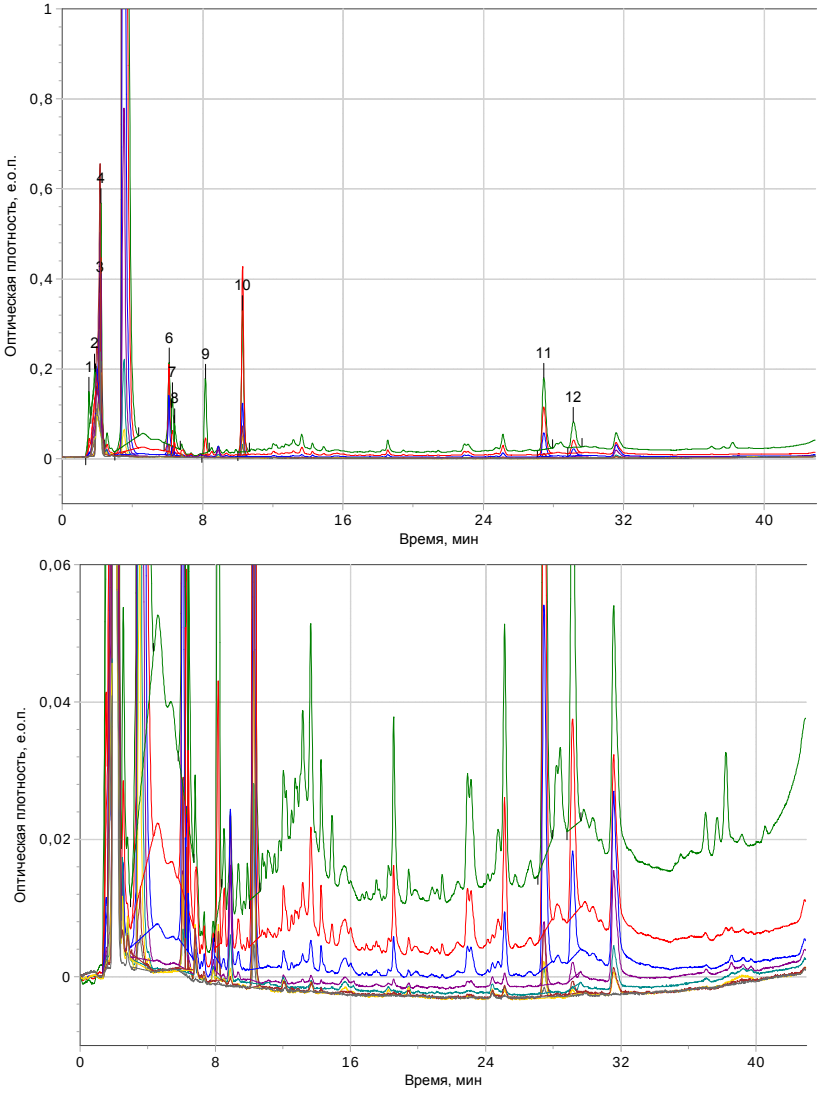


Рис. 3. Хроматограмма сыворотки крови здорового человека при малом и большом увеличении

2.1. Предварительная обработка сигнала

Достоверность любого результата будет приближаться к нулю, если уровень спектральных шумов будет иметь тот же порядок, что и максимальное поглощение на тех же длинах волн. В литературе [3] описаны эксперименты, позволяющие с уверенностью сказать, что относительный уровень шумов менее 2% не оказывает существенного влияния на результат исследования.

Фильтрация может понизить уровень шумов спектра за счет их сглаживания. Влияние того или иного фильтра на результат зависит от его вида. Например, при использовании метода изучения коэффициента корреляции семиразрядный фильтр Савицкого-Голея дает небольшое улучшение [3, 13]. Его влияние возрастает при расширении полосы фильтрации. Однако если брать фильтр со слишком широкой полосой, то может потеряться тонкая структура данных.

Вообще, шум – т.е. нежелательный сигнал детектора – может быть по своей природе как электронным, так и химическим [1]. Шум базовой линии, который мы видим – это то, что остается после фильтрации и сглаживания, убирающих высокие частоты. Разумеется, полностью избавиться от шума нельзя: если его частота близка к частотам известных хроматографических пиков, то фильтры могут удалить эти пики вместе с шумом.

Шум базовой линии размывает основание пика и затрудняет обнаружение начала и конца и вычисление площади пика. Шум в районе вершины пика вызывает ошибки при разделении методом долин, создавая «микропики».

Шум создает определенные ограничения на минимальное количество вещества, которое может быть распознано как отдельный пик. Слишком маленькие пики будут скрыты шумом базовой линии, и извлечь их оттуда будет невозможно. Минимальный полностью распознанный пик позволяет ввести понятие отношения сигнал/шум, которое демонстрирует связь между высотой пика и окружающим этот пик шумом. Руководства ACS (American Cancer Society) 1980 года определяют два предела: предел детектирования (сигнал/шум = 3), описывающий наименьший пик, который можно выделить из шума, и предел количественного определения (сигнал/шум = 10), описывающий наименьший пик, параметры которого могут быть измерены с достаточной точностью однако надо понимать, что понятие «достаточная точность» довольно расплывчатое, и для каждого конкретного анализа может потребоваться дополнительная корректировка пределов.

К сожалению, подобрать систему фильтров, подходящую для любых типов входных данных, чрезвычайно сложно. Опытным путем мы устано-

вили, что при обработке хроматограмм сыворотки крови наилучшие результаты достигаются при использовании вейвлет-преобразования Хаара и фильтра Савицкого–Голея с окном от 7 до 15 точек. Кроме того, для экспериментов с другими входными данными реализованы фильтр Гаусса и медианный фильтр, часто применяемые при работе с диодно-матричными детекторами.

Для построения вейвлета размера N используется функция Хаара

$$h_k(t) = \begin{cases} 2^{p/2}, \text{ если } (q-1)/2^p \le t < (q-0,5)/2^p, \\ -2^{p/2}, \text{ если } (q-0,5)/2^p \le t < q/2^p \quad (k = 0,1,...,N-1), \\ 0 \text{ иначе}, \end{cases}$$

определенная на интервале $0 \le t \le 1$. Параметр p определяется из соотношений $2^p < k$ и $2^{p+1} \ge k$, параметр q равен $k - 2^p + 1$. Матрица вейвлета строится из значений функции $h_k(t)$ при $t = \dfrac{m}{N}$ $(m = 0,1,\ldots,N-1)$.

Сглаживающий фильтр Савицкого–Голея позволяет снизить уровень шума не внося значительных искажений в площадь пиков. Значение сглаживающей функции в точке x_m вычисляется по формуле

$$SG_N(x_m) = \sum_{i=-\frac{N-1}{2}}^{\frac{N-1}{2}} \frac{p_{N,i} \cdot x_{m+i}}{h_N},$$

где N – ширина окна фильтра (имеет вид $2k+1$), а p_N и h_N – заранее известные весовые параметры:

$p_5 = \{17,12,-3\}$, $h_5 = 35$,

$p_7 = \{7,6,3,-2\}$, $h_7 = 21$

и т.д.

Медианный фильтр является одним из нелинейных фильтров с конечной импульсной характеристикой. Значения отсчетов внутри окна фильтра сортируются в порядке возрастания (убывания); и значение, находящееся в середине упорядоченного списка, поступает на выход фильтра. В случае четного числа отсчетов в окне выходное значение фильтра равно среднему значению двух отсчетов в середине упорядоченного списка. Затем окно перемещается вдоль фильтруемого сигнала, и вычисления повторяются.

2.2. Разбиение

Разбиение отфильтрованной хроматограммы на отдельные пики является наиболее ресурсоемкой частью анализа. Эта операция включает в себя нахождение начала, вершины и конца пика и построение базовой линии, ограничивающей его снизу.

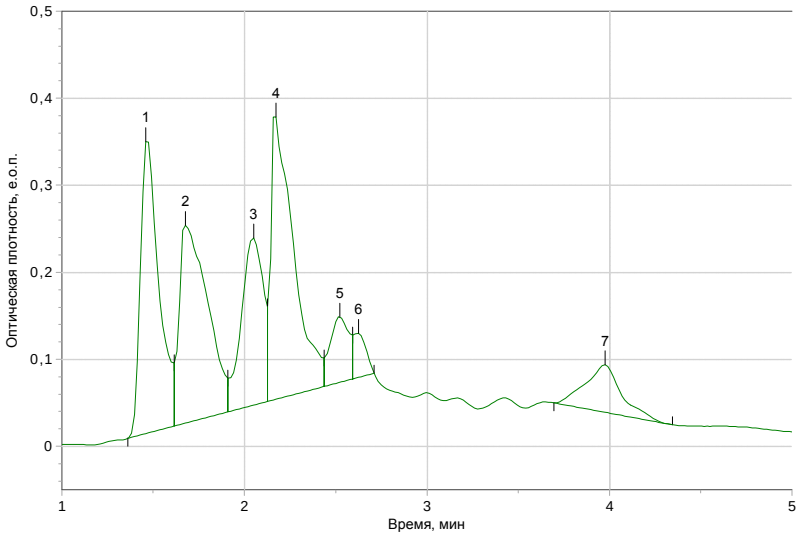


Рис. 4. Фрагмент хроматограммы, разбитой на отдельные пики с ортогональной базовой линией

Перед тем, как приступить к собственно разбиению, необходимо определить несколько важных параметров, описывающих само понятие пика – минимальные ширину, высоту и площадь, позволяющие отделить значимые данные от шума и дрейфа базовой линии (см. рис. 5).

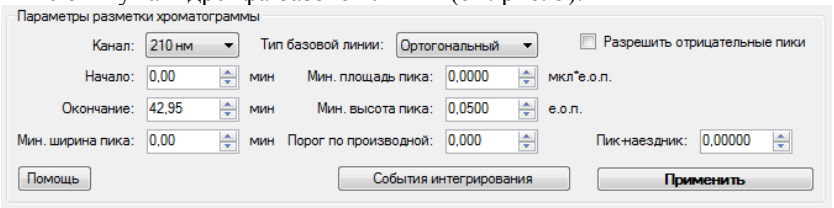


Рис. 5

Мы используем хорошо зарекомендовавший себя алгоритм детектирования пиков [4], основанный на изучении поведения первой производной хроматографической кривой. Для определения того, является ли наклон в данной точке значимым, величина производной делится на величину шума базовой линии, который определяется специальным алгоритмом на всей хроматограмме. Наклон считается значимым тогда, когда вычисленное отношение превышает некоторую заданную величину, называемую порогом срабатывания. Не полностью разделенные пики делятся по прямой (перпендикуляром к нулевой линии либо тангенциальным спуском). Пример результата работы алгоритма приведен на рис. 3. В случае, если система работает не в полностью автоматическом режиме, возможна ручная корректировка получившегося разбиения пользователем.

Одной из наиболее сложных проблем, возникающих при разбиении хроматограммы на компоненты, является разделение перекрывающихся пиков. В ряде случаев эта задача может быть решена путем анализа поведения спектральных отношений. В системе реализованы базовые алгоритмы такого анализа, позволяющие оценить степень перекрывания и разделить пики:

$$P_{1i} = x_i \cdot \frac{r_2 - x_i / y_i}{r_2 - r_1},$$

$$P_{2i} = x_i \cdot \left(1 - \frac{r_2 - x_i / y_i}{r_2 - r_1} \right),$$

где P_{ki} – значение оптической плотности пика k в точке i, x_i и y_i – измеренные значения оптической плотности для текущей и опорной длин волн в точке i, r_1 и r_2 – спектральные отношения для первого и второго пика, соответственно. Значения r_1 и r_2 задаются как x_i / y_i в точках, где перекрытие пиков отсутствует.

2.3. Сравнение компонентов

На текущий момент нашей основной задачей является разработка метода качественного сравнения полученных наборов пиков для получения ответа на вопрос: в чем именно заключается сходство и различие двух хрома-

тограмм? В качестве предварительного решения предлагается последовательное сравнение времен удерживания и спектральных векторов пиков.

На первом шаге алгоритма обе хроматограммы разбиваются на пики, после чего строится таблица соответствия:

Таблица 1. Таблица соответствия пиков

	Пик 1	Пик 2	Пик 3
Пик A	(2,1 мин; 6,2 °)	(1,5 мин; 11,9°)	(2,0 мин; 0,1°)
Пик B	(0,4 мин; 0,7°)	(0,1 мин; 0,4°)	(3,1 мин; 8,0°)
Пик C	(0,3 мин; 0,2°)	(3,0 мин; 4,1°)	(0,2 мин; 0,2°)

Пики первой хроматограммы обозначаются через A, B, и C, пики второй хроматограммы – через 1, 2 и 3. В ячейках таблицы вписаны расстояния по оси времени между вершинами соответствующих пиков и углы их спектральных отклонений.

После этого, из таблицы удаляются ячейки, в которых разница по времени или спектру превышает заданные экспертом величины (см. табл. 2).

Таблица 2. Таблица соответствия пиков после удаления невозможных кандидатов. В качестве пороговых значений были выбраны 2,0 мин и 1,0°

	Пик 1	Пик 2	Пик 3
Пик A			(2,0 мин; 0,1°)
Пик B	(0,4 мин; 0,7°)	(0,1 мин; 0,4°)	
Пик C	(0,3 мин; 0,2°)		(0,2 мин; 0,2°)

На последнем этапе выделяются наилучшие кандидаты. Для этого в таблице выбирается запись с минимальным углом отклонения (Пик A – Пик 3 в примере на табл. 3), помечается как наилучший кандидат и вычеркивается вместе со всей строкой и всем столбцом. Затем операция повторяется до тех пор, пока в таблице не останется записей. Таким образом строится словарь совпадений пиков двух хроматограмм, после чего можно сравнивать площади этих пиков, строить кластеры, и т.д.

Таблица 3. Таблица соответствия пиков с выделенными наилучшими кандидатами

	Пик 1	Пик 2	Пик 3
Пик A			(2,0 мин; 0,1°)
Пик B	(0,4 мин; 0,7°)	(0,1 мин; 0,4°)	
Пик C	(0,3 мин; 0,2°)		(0,2 мин; 0,2°)

2.4. Вспомогательные функции

В системе реализовано несколько дополнительных функций, значительно облегчающих работу пользователя. Одной из них является возможность градуировок, позволяющих вычислять концентрацию того или иного вещества в образце, основываясь на данных об объеме пробы, площади соответствующего пика и заранее заданных эталонных концентрациях (рис. 6). Аппроксимация производится с использованием метода наименьших квадратов и имеет вид $Q = k \cdot A$ или $Q = k \cdot A + b$, где Q – количество вещества, и A – площадь пика.

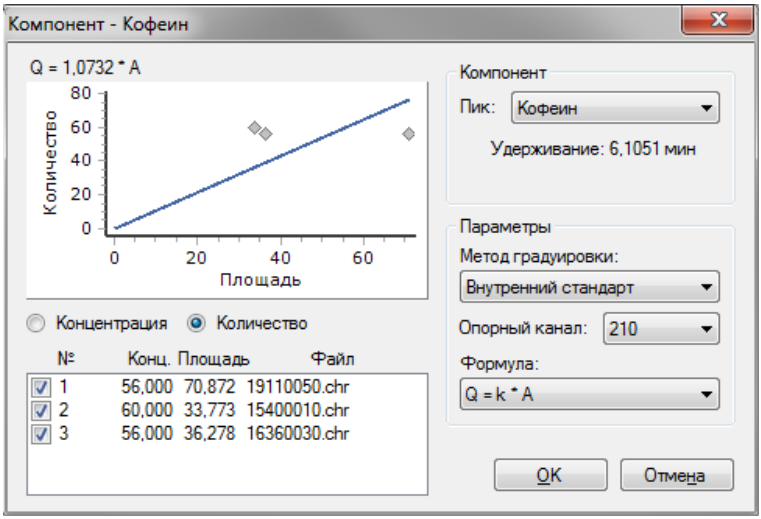


Рис. 6. Диалог выбора градуировочных параметров. На графике изображена аппроксимированная зависимость между площадью пика и количеством вещества (прямая) и результаты трех градуировочных экспериментов (точки)

Кроме того, предусмотрена возможность проверки корректности работы хроматографа. После анализа специально созданной пробы выполняется обработка полученной хроматограммы, и измеряются определенные параметры. Если значения этих параметров отклоняются от эталонных менее чем на заданную величину – хроматограф считается годным для дальнейшей работы с базой данных. Пример отчета по валидации приведен на рис. 7.

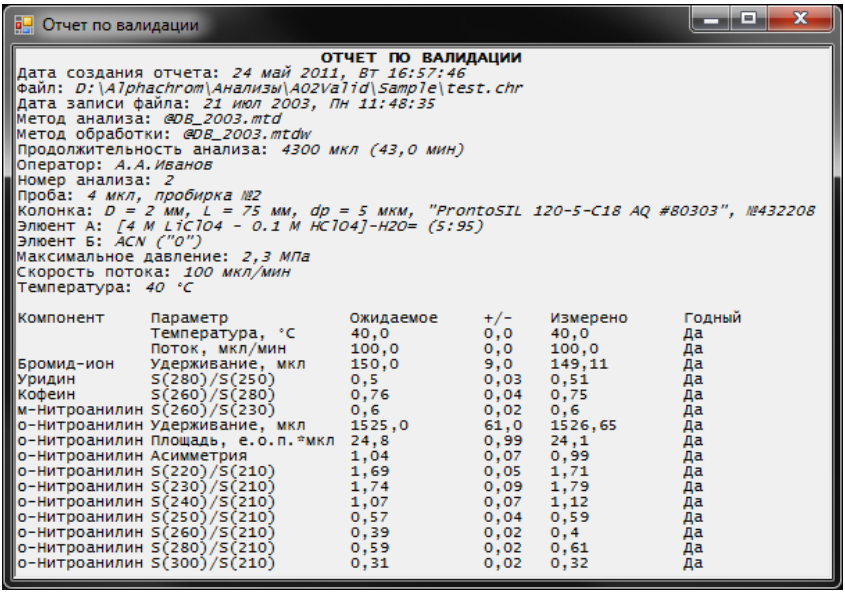


Рис. 7. Пример отчета по валидации хроматографа.
Все измеренные параметры не выходят за рамки допустимых погрешностей

3. РЕЗУЛЬТАТЫ

Разрабатываемая нами система помогает выделить химические соединения, характерные для какого-либо определенного заболевания. Имеющиеся на данный момент алгоритмы обеспечивают приемлемую точность, однако, к сожалению, накладывают некоторые ограничения на методы подготовки пробы и проведения хроматографического анализа – на приготовление пробы сейчас уходит около суток.

Текущим результатом работы стала готовая легко расширяемая программная платформа, обеспечивающая удобный доступ к хроматографическим данным, их обработку базовыми алгоритмами и визуализацию результатов анализа. Алгоритмы, реализованные на данный момент, позволяют проводить фильтрацию входных данных, кластеризацию (как в автоматическом режиме, так и вручную), идентификацию веществ по базе спектральных данных, выполнять процедуру валидации хроматографа и обрабатывать хроматограммы в пакетном режиме.

Также немаловажным, хотя и побочным результатом проводимого исследования стало приложение, позволяющее моделировать хроматограммы с асимметричными пиками (описываемые гауссианой, модифицированной экспоненциальной функцией – EMG), и эмулятор хроматографа «Милихром А-02» («Тренажер “Жидкостный хроматограф”»), используемый сейчас на кафедре аналитической химии ФЕН НГУ.

СПИСОК ЛИТЕРАТУРЫ

1. Dyson N. Chromatographic Integration Methods. – Letchworth, UK: Royal Society of Chemistry, 1998.
2. Dolan J. Integration Problems // LCGC North America. – 2009. – № 27(10).
3. Л. Хубер. Применение диодно-матричного детектирования в ВЭЖХ. – М.: Мир, 1993.
4. МультиХром для Windows 9x & NT, версия 1.5x-E. Руководство пользователя. – Новосибирск: ЗАО Институт хроматографии Эконова, 1997.
5. Raymond P. W. Scott. Liquid Chromatography Detectors // Library for Science, LLC. – 2003.
6. Цвет М.С. О новой категории адсорбционных явлений и о применении их к биохимическому анализу // Труды Варшавского общества естествоиспытателей 1903 года, Отделение биологии. – Протокол №6. – С. 1–20.
7. Heyden Y.V. Extracting Information from Chromatographic Herbal Fingerprints / LCGC Europe. – September 2008. – С. 438–443.
8. Померанцев А.Л., Родионова О.Е. Хемометрика в аналитической химии [Электронный ресурс] / А.Л. Померанцев, О.Е. Родионова. Режим доступа: http://www.chemometrics.ru. Дата обращения: 20.11.2011.
9. Zeng Z-D., Liang Y-Z., Xu C-J. Comparing chemical fingerprints of herbal medicines using modified window target-testing factor analysis // Anal. Bioanal. Chem. – 2005.– № 381. – С. 913–924.
10. Hansen P.W. Pre-processing method minimizing the need for reference analyses / J.Chemom. – 2001. – № 15. – С. 123.

11. Померанцев А.Л. Методы нелинейного регрессионного анализа для моделирования кинетики химических и физических процессов : Дис. д-ра физ.-мат. наук / А.Л. Померанцев ; Москва, ИХФ РАН, 2003.
12. Азарова И.Н., Барам Г.И., Гольдберг Е.Л. Предсказание объемов удерживания и УФ-спектров пептидов в обращенно-фазовой ВЭЖХ // Биоорганическая химия. – 2006. – №1(32) . – С.56–63.
13. Savitzky A., Golay M.J.E. Smoothing and differentiation of data by simplified least squares procedures // An. Chem. – 1964. – № 12.
14. Свидетельство № 38-03 об аттестации МВИ. Хроматографические и спектральные параметры УФ-поглощающих веществ. Методика выполнения измерений методом высокоэффективной жидкостной хроматографии.
15. Свидетельство № 67-06 об аттестации МВИ. Массовая концентрация УФ-поглощающих веществ. Методика выполнения измерений методом высокоэффективной жидкостной хроматографии.

Д.И. Бушин, И.Б. Вирбицкайте

ТРАССОВЫЕ ЭКВИВАЛЕНТНОСТИ ВРЕМЕННЫХ СЕТЕЙ ПЕТРИ*

1. ВВЕДЕНИЕ

Поведенческие эквивалентности обычно используются при спецификации и верификации систем с целью сравнения их поведения, а также упрощения их структуры. В теории параллельных систем и процессов известно большое разнообразие поведенческих эквивалентностей, взаимосвязи между которыми хорошо изучены в литературе (см., например, [2, 6]). Можно выделить два критерия классификации семантик, относительно которых определяются и исследуются модели и эквивалентности параллельных недетерминированных систем.

Первый критерий — степень точности, с которой учитываются точки недетерминированного выбора альтернативных действий системы. На основе этого критерия был сформирован так называемый спектр семантик «линейное/ветвистое время». Типичным представителем семантики линейного времени является *трассовая эквивалентность*. При трассовом подходе сравниваются поведения систем, представленные в виде множеств последовательностей действий, выполняемых системами, — языков систем.

На основе второго критерия классификации семантик был построен так называемый спектр «интерливинг/частичный порядок». Семантики различаются по степени, с которой учитывается отношение причинной зависимости между действиями системы, представленное частичным порядком, причем отсутствие частичного порядка между действиями системы означает, что эти действия параллельны. В интерливинговой семантике выполнение системы моделируется последовательностью выполняемых действий, не отражающей явно их причинную зависимость. Было сделано много попыток выйти за пределы интерливингового подхода, чтобы позволить внешнему наблюдателю с помощью эквивалентностей различать системы, учитывая параллелизм, используемый в их вычислениях. Как результат, в литературе появилось множество экви-

*Данная работа выполнена при поддержке DFG-РФФИ (грант No 436 RUS 113/1002/01, грант No 09-01-91334)

валентностей, основанных на моделировании причинной зависимости с помощью частичных порядков (см., например, [4]).

Известно, что анализ поведения параллельных систем реального времени — сложная задача, решение которой невозможно без привлечения формальных методов и средств. С этой целью за последнее десятилетие были разработаны различные модели, учитывающие временные характеристики функционирования систем, — временные автоматы, временные сети Петри, временные структуры событий и т.д. Понятие времени также было введено и в поведенческие эквивалентности. Иерархия взаимосвязей временных эквивалентностей в семантиках «интерливинг/частичный порядок» и «линейное/ветвистое время» в контексте локальных структур событий с непрерывным временем построена в статье [3].

В данной работе определяется и исследуется семейство трассовых эквивалентностей в семантиках «интерливинг/частичный порядок» в контексте временных сетей Петри. Изучаемые эквивалентности основываются на понятии временного процесса, т.е. временного расширения причинной сети за счет сопоставления глобальных моментов времени срабатываниям переходов. При этом рассматриваются только корректные по времени процессы, т.е. такие, временная функция которых удовлетворяет специально разработанным свойствам корректности. Устанавливаются взаимосвязи эквивалентностей, и строится иерархия классов эквивалентных временных сетей Петри.

Статья организована следующим образом. В разделе 2 вводятся основные понятия и обозначения, связанные со структурой и поведением временных сетей Петри (ВСП). В разделе 3 рассматриваются временные процессы ВСП, и изучаются их свойства. Определения эквивалентностей ВСП и исследование их взаимосвязей представлены в разделе 4.

2. ВРЕМЕННЫЕ СЕТИ ПЕТРИ

В данном разделе рассматриваются базовые определения, связанные со структурой и поведением временной сети Петри [5].

Пусть $\mathbb{N}$ — множество натуральных чисел, и $\mathbb{R}$ — множество действительных чисел. Определим множество $\mathbf{Interv} = \{[d_1, d_2 \subset \mathbb{R} \mid d_1 \leq d_2 \ \& \ d_1, d_2 \in \mathbb{N}\}$. Пусть Act — множество действий.

Определение 2.1. *Временная сеть Петри* (ВСП) — это набор $TN = (N = (P,\ T,\ F,\ M_0,\ L),\ D)$, где $N = (P, T, F, M_0, L)$ — *(помеченная)*

базовая сеть Петри (СП) с конечным множеством P мест, конечным множеством T переходов ($P \cap T = \emptyset$), отношением инцидентности $F \subseteq (P \times T) \cup (T \times P)$, начальной разметкой $M_0 \subseteq P$, помечающей функцией $L : T \to Act$, сопоставляющей каждому переходу $t \in T$ действие $L(t) \in Act$, и $D : T \longrightarrow \mathbf{Interv}$ — *статическая временная функция*, сопоставляющая каждому переходу $t \in T$ временной интервал $D(t) \in \mathbf{Interv}$.

Для элемента $x \in P \cup T$ ${}^\bullet x = \{y \mid y\ F\ x\}$ — множество его входных элементов, а $x^\bullet = \{y \mid x\ F\ y\}$ — множество его выходных элементов. Будем считать, что для каждого перехода $t \in T$ верно, что $|{}^\bullet\ t\ | > 0$ и $|\ t^\bullet\ | > 0$. Если $D(t) = [d_1, d_2]$, то через $Eft(t) = d_1$ и $Lft(t) = d_2$ будем обозначать соответственно раннее и позднее времена срабатывания перехода t.

Разметка M ВСП TN определяется как произвольное подмножество $M \subseteq P$ мест. Переход $t \in T$ *готов сработать при разметке* M (обозначается $M \xrightarrow{t}$), если ${}^\bullet t \subseteq M$. Пусть $En(M)$ — множество всех переходов, готовых сработать при разметке M.

Состояние ВСП TN — это пара $S = (M, I)$, где M — разметка и $I : En(M) \longrightarrow \mathbb{R}$ — *динамическая временная функция* переходов из $En(M)$. *Начальное состояние* — это пара $S_0 = (M_0, I_0)$, где $I_0(t) = 0$ для всех t из $E_n(M_0)$. Переход t *готов сработать в состоянии* $S = (M, I)$ *в относительный момент времени* θ, если верно:

1. $t \in En(M)$.
2. $(M \backslash {}^\bullet t) \cap t^\bullet = \emptyset$.
3. $Eft(t) \leq I(t) + \theta$.
4. $\forall t' \in En(M) \circ I(t') + \theta \leq Lft(t')$.

Будем говорить, что переход t *находится в контакте* в состоянии S, если для него выполнены условия 1, 3 и 4, но не выполнено условие 2. Пусть $Contact(S)$ обозначает множество всех переходов, находящихся в контакте в состоянии S.

Если переход t готов сработать в состоянии $S = (M, I)$ в относительный момент времени θ, то его срабатывание меняет состояние S на новое состояние $S' = (M', I')$ (обозначается $S \overset{(t,\theta)}{\to} S'$) по следующему правилу:

- $\widehat{M} = M \backslash {}^\bullet t$,
- $M' = \widehat{M} \cup t^\bullet$,

$$\bullet\ \forall t' \in T \circ I'(t') = \begin{cases} I(t) + \theta, & \text{если } t \in En(\widehat{M}), \\ 0, & \text{если } t' \in En(M') \backslash En(\widehat{M}), \\ \text{не определено}, & \text{иначе.} \end{cases}$$

Последовательность $S_0 \overset{(t_1,\theta_1)}{\to} S_1 \ \ldots\ S_{n-1} \overset{(t_1,\theta_n)}{\to} S_n$ $(n \geq 0)$ называется *последовательностью срабатываний* ВСП TN. Состояние S ВСП TN называется *достижимым*, если существует последовательность срабатываний, приводящая в состояние S. Пусть $RS(TN)$ обозначает множество достижимых состояний ВСП TN.

Будем говорить, что ВСП TN является

- *свободной от контактов*, если для каждого $S \in RS(TN)$ верно, что $Contact(S) = \emptyset$,
- *прогрессирующей по времени*, если для всякого множества переходов $\{t_1, t_2, \ldots, t_n\}$ таких, что $t_i^\bullet \cap^\bullet t_{i+1} \neq \emptyset$ и $t_n^\bullet \cap^\bullet t_1 \neq \emptyset$ для каждого $1 \leq i < n$, верно, что $\sum_{1 \leq i \leq n} Eft(t_i) > 0$.

В дальнейшем будем рассматривать только свободные от контактов и прогрессирующие по времени ВСП.

3. ВРЕМЕННЫЕ ПРОЦЕССЫ ВСП

Сначала введем понятие сети. Тройка (B, E, G) называется *сетью*, если $B \neq \emptyset$ — множество условий, $E \neq \emptyset$ — множество событий $(E \cap B = \emptyset)$, $G \subseteq (B \cup E) \times (E \cup B)$ — отношение инцидентности такое, что $\{x \mid (x, y) \in G\} \cup \{y \mid (x, y) \in G\} = E \cup B$. Для произвольного элемента $x \in B \cup E$ через ${}^\bullet x = \{y \mid (y, x) \in G\}$ и $x^\bullet = \{y \mid (x, y) \in G\}$ будем обозначать множества его входных и выходных элементов соответственно.

Рассмотрим понятие (помеченной) С-сети. Пара $C = (N, l)$ называется *(помеченной) С-сетью*, если $N = (B, E, G)$ — сеть такая, что

- $\preceq = G^*$ — частичный порядок[1] (*отношение причинной зависимости*),
- $\forall x \in (B \cup E)\ \circ \downarrow x = \{y \in (B \cup E) \mid y \preceq x\}$ — конечное множество,
- $\forall b \in B\ \circ\ |{}^\bullet b| \leq 1\ \wedge\ |b^\bullet| \leq 1$,

и $l : E \to Act$ — функция пометки, сопоставляющая каждому событию $e \in E$ действие $l(e) \in Act$. Множества входных и выходных условий С-сети C будем обозначать соответственно ${}^\bullet C = \{b \in B \mid {}^\bullet b = \emptyset\}$ и $C^\bullet = \{b \in B \mid b^\bullet = \emptyset\}$. Компоненты С-сети C будем писать с нижним индексом C. Для произвольного левозамкнутого относительно $\preceq_C$ подмножества событий $E' \subseteq E_C$ определим множество $Cut(E') = (E'^\bullet \bigcup {}^\bullet C) \backslash {}^\bullet E'$.

[1] Антисимметричность исключает циклы.

Пусть $C = (B, E, G, l)$ и $C' = (B', E', G', l')$ — С-сети. Отображение $\beta : B \cup E \to B' \cup E'$ — изоморфизм между C и C' (обозначается $\beta : C \simeq C'$), если верно:

- β — биективное отображение, такое, что $\beta(B) = B' \wedge \beta(E) = E'$,
- $\forall x, y \in B \cup E \ \diamond \ G(x, y) = G'(\beta(x), \beta(y))$,
- $\forall e \in E \ \diamond \ L(e) = L'(\beta(e))$.

С-сети C и C' изоморфны (обозначается $C \simeq C'$), если существует изоморфизм $\beta : C \simeq C'$.

Рассмотрим понятие процесса ВСП TN.

Определение 3.1. Пусть $TN = (N = (P, T, F, M_0, L), D)$ — ВСП. Тогда $\rho = (C = (B, E, G, l), \phi)$ — *процесс* ВСП TN, если $\phi : B \cup E \to P \cup T$ — гомоморфизм, удовлетворяющий свойствам:

- $\phi(B) \subseteq P$ и $\phi(E) \subseteq T$,
- $\forall e \in E \ \diamond \ \phi({}^\bullet e) = {}^\bullet \phi(e) \ \wedge \ \phi(e^\bullet) = \phi(e)^\bullet$,
- $\forall e \in E \ \diamond \ l(e) = L(\phi(e))$.

Пусть $\rho = (C, \phi)$ и $\rho' = (C', \phi')$ — процессы ВСП TN и TN', соответственно. Отображение $\beta : B_C \cup E_C \longrightarrow B_{C'} \cup E_{C'}$ — изоморфизм между ρ и ρ' (обозначается $\beta : \rho \simeq \rho'$), если $\beta : C \simeq C'$ и $\forall x \in B_C \cup E_C \ \diamond \ \phi(x) = \phi'(\beta(x))$. Процессы ρ и ρ' изоморфны (обозначается $\rho \simeq \rho'$), если существует изоморфизм $\beta : \rho \simeq \rho'$.

Процесс $\rho_0 = (C_0, \phi_0)$ ВСП TN называется *начальным*, если $M_0 = \phi_0({}^\bullet C_0)$ и $E_{C_0} = \emptyset$. Будем говорить, что в ВСП TN процесс $\rho = (C, \phi)$ *допустим после процесса* $\rho' = (C', \phi')$, если $\phi({}^\bullet C) = \phi(C'^\bullet)$. Для ВСП TN множество всех ее процессов, допустимых после процесса ρ, обозначим через $\mathcal{P}(\mathcal{TN}, \rho)$, и множество всех ее процессов, допустимых после начального процесса, — через $\mathcal{P}(\mathcal{TN})$.

Пусть $\rho = (C, \phi), \rho' = (C', \phi') \in \mathcal{P}(\mathcal{TN})$ и $\widehat{\rho} = (\widehat{C}, \widehat{\phi}) \in \mathcal{P}(\mathcal{TN}, \rho)$. Тогда процесс ρ — *префикс* процесса ρ', если $E_C \subseteq E_{C'}$ — левозамкнутое относительно $\preceq_{C'}$ и $\phi = \phi'|_{E_C}$. Процесс $\widehat{\rho}$ — *суффикс* процесса ρ', если $E_{\widehat{C}} = E_{C'} \backslash E_C$ и $\widehat{\phi} = \phi'|_{\widehat{E}}$. Тогда ρ' — *расширение* ρ на процесс $\widehat{\rho}$, а $\widehat{\rho}$ — *расширяющий* процесс для ρ (обозначается $\rho \xrightarrow{\widehat{\rho}} \rho'$). Будем писать $\rho \longrightarrow \rho'$, если существует $\widehat{\rho}$, такой, что $\rho \xrightarrow{\widehat{\rho}} \rho'$.

Приведем определение временного процесса ВСП TN.

Определение 3.2. *Временной процесс* ВСП TN — это пара $\pi = (\rho, \tau)$, где $\rho = (C, \phi)$ — процесс ВСП TN и $\tau : E \to \mathbb{R}$ — временная функция, сопоставляющая каждому событию $e \in E$ глобальное время $\tau(e) \in \mathbb{R}$ его выполнения. *Длительность* π равна $time(\pi) = max\{\tau_\pi(e) \mid e \in E_\pi\}$.

Пусть $\pi = (\rho = (C, \phi), \tau)$ и $\pi' = (\rho' = (C', \phi'), \tau')$ — временные процессы ВСП TN и TN' соответственно. Отображение $\beta : B_C \cup E_C \to B_{C'} \cup E_{C'}$ — изоморфизм между π и π' (обозначается $\beta : \pi \simeq \pi'$), если $\beta : \rho \simeq \rho'$ и $\forall x \in E_C \ \diamond \ \tau(x) = \tau'(\beta(x))$. Временные процессы π и π' изоморфны (обозначается $\pi \simeq \pi'$), если существует изоморфизм $\beta : \pi \simeq \pi'$.

Начальный временной процесс ВСП TN — это пара $\pi_0 = (\rho_0, \emptyset)$, где ρ_0 — начальный процесс ВСП TN. Будем говорить, что в ВСП TN временной процесс $\pi = (\rho, \tau)$ *допустим после временного процесса* $\pi' = (\rho', \tau')$, если ρ допустим после ρ' и $\tau(e) \geq time(\pi')$ для всех $e \in E_C$. Для ВСП TN множество всех ее временных процессов, допустимых после временного процесса π, обозначим через $\mathcal{TP}(\mathcal{TN}, \pi)$, и множество всех ее временных процессов, допустимых после начального временного процесса, — через $\mathcal{TP}(\mathcal{TN})$.

Пусть $\pi = (\rho, \tau) \in \mathcal{TP}(TN, \pi')$. Если $B' \subseteq B_C$ и $t \in En(\phi(B'))$, то глобальный момент времени, когда во всех входных местах перехода t появляются фишки, определяется следующим образом:

$$\mathbf{TOE}(B', t, \pi') = \max(\{\tau(^\bullet b) \mid b \in B' \setminus^\bullet C \ \wedge \ \phi(b) \in^\bullet t\} \cup \{time(\pi')\}).$$

Для $\pi = (\rho, \tau) \in \mathcal{TP}(TN, \pi')$ функция τ называется *корректным таймированием*, если для каждого $e \in E_C$ верно:

- $\tau(e) \geq \mathbf{TOE}(^\bullet e, \phi(e), \pi') + Eft(\phi(e))$,
- $\forall t \in En(\phi(C_e)) \diamond \tau(e) \leq \mathbf{TOE}(C_e, t, \pi') + Lft(t)$,
  где $C_e = Cut(Earlier(e))$ и $Earlier(e) = \{e' \in E_C \mid \tau(e') < \tau(e)\}$.

Временной процесс $\pi = (\rho, \tau) \in \mathcal{TP}(TN, \pi')$ называется *корректным*, если τ — корректное таймирование. В дальнейшем будем рассматривать только корректные временные процессы.

Пусть $\pi = (\rho, \tau)$, $\pi' = (\rho', \tau') \in \mathcal{TP}(TN)$ и $\widehat{\pi} = (\widehat{\rho}, \widehat{\tau}) \in \mathcal{TP}(TN, \pi)$. Тогда временной процесс π' — *расширение* временного процесса π на временной процесс $\widehat{\pi}$, а $\widehat{\pi}$ — *расширяющий* временной процесс для π (обозначается $\pi \xrightarrow{\widehat{\pi}} \pi'$), если $\rho \xrightarrow{\widehat{\rho}} \rho'$ и $\tau = \tau'|_{E_C}$, $\widehat{\tau} = \tau'|_{\widehat{E}_{\widehat{C}}}$.

Для $\pi = (\rho, \tau)$, $\pi' = (\rho', \tau') \in \mathcal{TP}(TN)$ π' — *расширение* π *на*:

- *действие, произошедшее в относительный момент времени* θ, (обозначается $\pi \xrightarrow{(a,\theta)} \pi'$), если существует расширяющий временной процесс $\widehat{\pi}$ для π такой, что $\widehat{E} = \{e\}$, $\widehat{\tau}(e) = time(\pi) + \theta$ и $\widehat{l}(e) = a$,

- *мультимножество A действий, произошедших в относительный момент времени* θ, (обозначается $\pi \xrightarrow{(A,\theta)} \pi'$), если существует расширяющий временной процесс $\widehat{\pi}$ для π такой, что $\widehat{\preceq} \cap (\widehat{E} \times \widehat{E}) = \emptyset$, $\widehat{l}(\widehat{E}) = A$ и $\widehat{\tau}(e) = time(\pi) + \theta$ для всех $e \in \widehat{E}$.

4. ТРАССОВЫЕ ЭКВИВАЛЕНТНОСТИ ВСП И ИХ ВЗАИМОСВЯЗИ

В этом разделе рассматриваются понятия поведенческих эквивалентностей ВСП, и исследуются из взаимосвязи.

Сначала определим вспомогательные понятия для ВСП TN.

- Слово $\omega = (a_1, \theta_1)...(a_n, \theta_n)$ из алфавита $Act \times \mathbb{R}$ называется *временным интерливинговым следом* ВСП TN, если в ней существует последовательность вида $\pi_0 \xrightarrow{(a_1,\theta_1)} \pi_1 \ \ldots \ \pi_{n-1} \xrightarrow{(a_n,\theta_n)} \pi_n$. Множество всех временных интерливинговых следов ВСП TN обозначим через $L_i(TN)$.
- Слово $\Omega = (A_1, \theta_1)...(A_n, \theta_n)$ из алфавита $\mathbb{N}_f^{Act} \times \mathbb{R}$ называется *временным шаговым следом* ВСП TN, если в нем существует последовательность вида $\pi_0 \xrightarrow{(A_1,\theta_1)} \pi_1 \ldots \pi_{n-1} \xrightarrow{(A_n,\theta_n)} \pi_n$. Множество всех временных шаговых следов TN обозначим через $L_s(TN)$.
- Класс изоморфизма временного процесса $\pi = (\rho, \tau) \in \mathcal{TP}(TN)$ называется *временным процессным следом* ВСП TN. Множество всех временных процессных следов ВСП TN обозначим через $L_{pr}(TN)$.

Следствие 1. Для любой ВСП TN верно $L_i(TN) \subseteq L_s(TN)$.

Определение 4.1. Пусть $* \in \{i, s, pr\}$. Тогда ВСП TN и TN' называются *-трассово эквивалентными (обозначается $TN \equiv_* TN'$), если $L_*(TN) = L_*(TN')$.

Теорема. Пусть $*, ** \in \{i, s, pr\}$. Для любых ВСП TN и TN' верно:

$$TN \equiv_* TN' \ \Rightarrow \ TN \equiv_{**} TN'$$

тогда и только тогда, когда в графе, изображенном на рис. 1(а), существует дуга от $\equiv_*$ к $\equiv_{**}$.

Доказательство. ($\Rightarrow$) Проверим истинность импликаций на рис. 1(а).

- Связь 1 ($\equiv_s \to \equiv_i$) устанавливается с помощью следствия 1.

- Связь 2 ($\equiv_{pr} \rightarrow \equiv_s$) следует из определений временных шаговых/процессных следов и изоморфизма временных процессов.

Заметим, что связь 3, изображенная на рис. 1(б), следует из предыдущих пунктов.

($\Rightarrow$) Докажем, что в графе на рис. 1(а) от одной эквивалентности к другой нельзя провести дополнительной дуги, такой, что в этом графе не существует пути от первой эквивалентности ко второй.

- На рис. 1(в) изображены ВСП TN_1 и TN_2, которые являются i-трассово эквивалентными, но не s-трассово эквивалентными, так как только в TN_2 действия a и b могут произойти параллельно в глобальный момент времени 0. Следовательно, связь 1* отсутствует.
- На рис. 1(г) изображены ВСП TN_2 и TN_3, которые являются s-трассово эквивалентными, но не pr-трассово эквивалентными, поскольку только в TN_3 действие a может причинно зависеть от действия b. Тогда связь 2* отсутствует.
- Снова рассмотрим ВСП TN_1 и TN_2, изображенные на рис. 1(в). Отсутствие дуги 3* в графе следует из импликации: $TN_1 \not\equiv_s TN_2 \Rightarrow TN_1 \not\equiv_{pr} TN_2$.

Как известно, количество дуг полного направленного графа с $N = 3$ вершинами равно $N * (N - 1) = 6$. Таким образом, мы рассмотрели все возможные случаи. $\square$

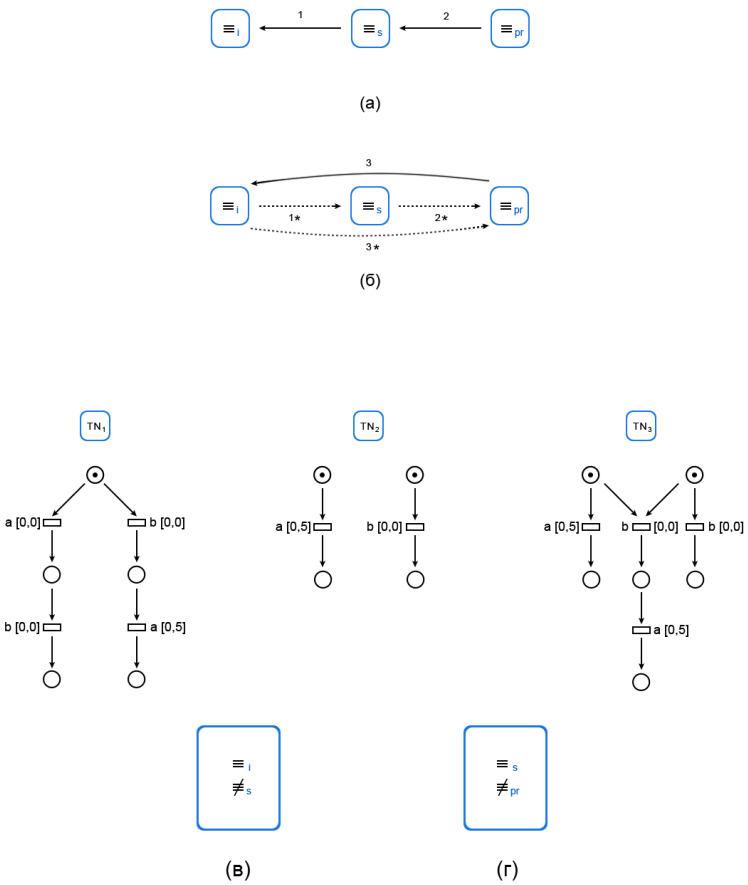


Рис. 1. Взаимосвязи эквивалентностей

СПИСОК ЛИТЕРАТУРЫ

1. Вирбицкайте И. Б. Сети Петри: модификации и расширения / Учебное пособие НГУ. — 2005. — 125 с.
2. Тарасюк И. В. Эквивалентности для поведенческого анализа параллельных и распределенных вычислительных систем. — Новосибирск: Академическое издание «Гео», 2007. — 224 с.
3. Andreeva M. V., Virbitskaite I. B. Observational Equivalences for Timed Stable Event Structures / Fundamenta Informaticae. — 2006. — Vol. 72. — P. 1-19.
4. van Glabbeek R.J., Goltz U. Refinement of actions and equivalence notions for concurrent systems / Acta Informatica. — 2001. — Vol. 37. — P. 229–327.
5. Merlin P., Faber D. J. Recoverability of communication protocols / IEEE Trans. of Communication. — 1976. — Vol. COM-24(9). — P. 183-195.
6. Pomello L., Rozenberg G., Simone C. A Survey of Equivalence Notions for Net Based Systems / Lecture Notes in Computer Science. — 1992. — Vol. 609. — P. 410-450.
7. Rosenberg G., Thiagarajan P. S. Petri Nets: Basic Notions, Structure, Behaviour / Lect. Notes Computer Science. — 1986. — Vol. 224. — P. 585-668.
8. Aura T., Lilius J.Time processes for time Petri nets / Lect. Notes Comput. Sci. – 1997. – Vol. 1248. – P. 136–155.

Н.Б. Зверев, С.А. Полетаев

О РЕАЛИЗАЦИИ АЛГОРИТМА ИЕРАРХИЧЕСКОГО КЛАСТЕРНОГО АНАЛИЗА НА GPU СРЕДСТВАМИ ТЕХНОЛОГИИ CUDA

1. ВВЕДЕНИЕ

В работе рассматривается алгоритм иерархической кластеризации [1, 2] и предлагается метод отображения данного алгоритма на параллельную мультипроцессорную систему, использующуюся на современных графических процессорах GPU. Для реализации алгоритма используется технология CUDA, разработанная компанией NVIDIA [7].

Заметим, что GPU предназначены для выполнения интенсивных расчётов. Задачи с интенсивным обращением к памяти или сложной логикой будут выполняться неэффективно, т.к. GPU обладает слабыми средствами кэширования при обращении к памяти и «не переносит» ветвлений в программе.

Модель вычислений CUDA, используемая на GPU, предполагает, что программист вначале разбивает задачу на независимые части (блоки), которые могут выполняться параллельно. Затем каждый блок разбивается на множество параллельно выполняющихся потоков (thread), которые могут зависеть друг от друга. CUDA обеспечивает средства расширения языка C++ для параллельного запуска множества потоков, выполняющих одну и ту же функцию (ядро, kernel). Максимальный размер ядра – 2 миллиона инструкций. Потоки объединяются в блоки (до 512 потоков), блоки объединяются в сетки (решётки, grid).

Сегодня можно сказать, что архитектура параллельных вычислений CUDA от NVIDIA позволила многим исследователям, применяя язык C++, задействовать вычислительные мощности графических процессоров для решения сложных расчетных задач. При этом привлекательно то, что эта аппаратура, в отличие от суперкомпьютеров, доступна многим по цене.

В работе в рамках некоторых естественных предположений делаются оценки времени выполнения алгоритма иерархической кластеризации в последовательном случае, параллельном случае для абстрактной параллельной машины и на GPU. Также получены соответствующие коэффициенты ускорения.

Процедура построения матрицы расстояний между кластерами реализована на GPU средствами системы CUDA, которая при больших размерностях задачи в десятки раз (более чем в 60) обгоняет по производительности ту же программу на C++ в случае ее реализации на центральном процессоре. Процедура поиска минимального элемента матрицы, значительно уменьшающая обмен данными между устройством и хостом, также реализована на GPU средствами системы CUDA. Алгоритм полностью реализован на компьютере, и приведены результаты его тестирования.

2. ОСНОВЫ КЛАСТЕРНОГО АНАЛИЗА

Кластерный анализ (англ. Data clustering) – задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из похожих объектов, а объекты разных кластеров существенно отличались. Практическое приложение такого рода можно видеть в [5].

Входные данные могут иметь различный вид. Первый вариант – использование признакового описания объектов. Каждый объект описывается набором (вектором) своих характеристик, называемых признаками. В общем случае признаки могут быть числовыми или нечисловыми [3,4]. Последние, как правило, тоже оцифровываются.

Второй вариант – когда задается матрица расстояний между объектами.

Цели кластеризации могут быть различные.

1. Понимание структуры данных. Разбиение множества данных на группы схожих объектов позволяет лучше понять структуру.
2. Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера.
3. Обнаружение новизны (англ. novelty detection). Выделяются нетипичные объекты, которые не удаётся присоединить ни к одному из кластеров.

Процедура иерархического кластерного анализа [1, 2] предусматривает группировку как объектов (строк матрицы данных), так и переменных (столбцов). В начале процесса кластеризации все объекты считаются отдельными кластерами, которые в ходе алгоритма объединяются. Процесс агрегирования можно представить в виде иерархического дерева.

Возникает горизонтальная древовидная диаграмма. Диаграмма начинается со всех объектов в классе, которые располагаются вертикально. По-

степенно можно «ослаблять» критерий того, какие объекты являются уникальными, а какие нет.

Другими словами, понижается порог, относящийся к решению об объединении двух или более объектов в один кластер.

В результате связывается всё большее и большее число объектов и агрегируется (объединяется) все больше и больше кластеров, состоящих из все более различающихся элементов. На последнем шаге все объекты объединяются вместе. На таких диаграммах горизонтальные оси обычно представляют расстояние между кластерами.

Успешный анализ позволяет выделить кластеры как отдельные ветви и дать им содержательную интерпретацию.

Расстояния (или, в общем случае, меры близости) между точками и между кластерами могут быть определены разными способами. Например, могут быть использованы: евклидова или чебышевская метрика, супремум-норма и др.

3. МЕТОДЫ КЛАСТЕРИЗАЦИИ

Таким образом, на первом шаге мы имеем метрику или меру близости между отдельными точками. Далее мы должны определить расстояние, или меру близости между кластерами, чтобы в более общем случае можно было продолжить процесс иерархической кластеризации. Для этого применяются различные методы.

1. Одиночная связь (метод ближайшего соседа). В этом методе расстояние между двумя кластерами определяется как расстояние между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах.

2. Полная связь (метод наиболее удаленных соседей). В этом методе расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. «наиболее удаленными соседями»).

3. Невзвешенное попарное среднее. В этом методе расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них.

4. Взвешенное попарное среднее. Метод идентичен методу невзвешенного попарного среднего за исключением того, что при вычислениях размер соответствующих кластеров (т.е. число объектов, содержащихся в них) используется в качестве весового коэффициента.

5. Невзвешенный центроидный метод. В этом методе расстояние между двумя кластерами определяется как расстояние между их центрами тяжести.

6. Взвешенный центроидный метод (медиана). Этот метод идентичен предыдущему за исключением того, что при вычислениях используются веса для учёта разницы между размерами кластеров (т.е. числами объектов в них).

7. Метод Варда. Этот метод отличается от всех других методов, поскольку он фактически использует методы дисперсионного анализа для оценки расстояний между кластерами. Метод минимизирует некоторую сумму квадратов расстояний. Более точно, в качестве расстояния между кластерами берется прирост суммы квадратов расстояний объектов до центров кластеров, получаемый в результате их объединения.

После выполнения очередного шага агломеративной процедуры необходимо убедиться, что желаемое разбиение достигнуто. Существуют различные методы определения критерия остановки процедуры: получено определенное заранее количество кластеров; все кластеры содержат более определенного числа элементов; кластеры обладают требуемым соотношением внутренней однородности и разнородности между собой и др.

4. ТЕХНОЛОГИЯ CUDA

Технология CUDA – это программно-аппаратная вычислительная архитектура NVIDIA [7], основанная на расширении языка C, которая даёт возможность организации доступа к набору инструкций графического ускорителя и управления его памятью при организации параллельных вычислений. CUDA помогает реализовывать алгоритмы, выполнимые на графических процессорах видеоускорителей GeForce восьмого поколения и старше (серии GeForce 8, GeForce 9, GeForce 200), а также Quadro и Tesla.

Хотя трудоёмкость программирования GPU при помощи CUDA довольно велика, она ниже, чем при использовании раннимх GPGPU решениям. Такие программы требуют разбиения приложения между несколькими мультипроцессорами подобно MPI программированию, но без разделения данных, которые хранятся в общей видеопамяти. И так как CUDA программирование для каждого мультипроцессора подобно OpenMP программированию, оно требует хорошего понимания организации памяти. Но, конечно же, сложность разработки и переноса на CUDA сильно зависит от приложения.

Набор для разработчиков содержит множество примеров кода и хорошо документирован. Процесс обучения требует около двух-четырёх недель для тех, кто уже знаком с OpenMP и MPI. В основе API лежит расширенный язык C, а для трансляции кода с этого языка в состав CUDA SDK входит компилятор командной строки nvcc, созданный на основе открытого компилятора Open64.

Программное обеспечение CUDA состоит из нескольких частей: драйвер аппаратуры, интерфейс программных приложений (API), две высокоуровневые математические библиотеки общего пользования CUFFT и CUBLAS, которые подробно описаны в [8, 9].

5. РЕАЛИЗАЦИЯ АЛГОРИТМА НА CUDA

5.1. Структура алгоритма

Алгоритм иерархического кластерного анализа является итерационным. Каждая итерация содержит несколько этапов: построение матрицы расстояний; поиск минимального элемента матрицы; объединение кластеров, расстояние между которыми минимальное.

Треугольная матрица расстояний $M = (m_{ij}), 0 \le i, j < N$ строится обходом по верхней части матрицы, т.е. по таким элементам m_{ij}, что $i > j$. В противном случае полагается $m_{ij} = 0$.

Пусть $M_N = M$. Далее при объединении кластеров возникает последовательность матриц расстояний $M_N, M_{N-1}, \ldots, M_{N-s}$. По ней строим соответствующую ей последовательность матриц $A_N, A_{N-1}, \ldots, A_{N-s}$, которые отражают принадлежность элементов кластерам.

$A_N = E$ – единичная матрица размерности $N \times N$.

Если A_k уже построена, то A_{k-1} строится следующим образом.

В матрице M происходит поиск такой пары (i, j), что $M\ (i, j) = m_{ij} = \min\{m_{kl} : 0 \le k, l < N\}$. Заметим, что может быть несколько пар (i, j), на которых достигается минимум по всем элементам матрицы. Тогда выбираем одну из них. Очевидно, что в этом случае соответствующие значения элементов матрицы равны.

Получив результат, «обходим» i-й и j-й столбцы матрицы A и заменяем элементы столбца i на дизъюнкцию соответствующих элементов столбцов i и j. Далее заменяем j-й столбец на k-й. При этом k должно быть не равно j. Вычеркиваем k-й столбец из матрицы т.е. при следующей итерации мы будем иметь матрицу M уже меньшего порядка по одной из размерностей; иначе говоря, количество кластеров уменьшится на один.

Заметим также, что в процессе k-й итерации мы работаем с матрицей $M_k = M_{N-k+1}$, т.е. элементов в этом случае $N'_k = (N_k^2 - N_k)/2$.

Далее аналогичным с [6] образом мы оценим эффективность алгоритма для различных ситуаций.

5.2. Оценка времени работы последовательного алгоритма

Шаг 1. Построение матрицы расстояний

В последовательном случае для построения матрицы расстояний необходимо вычислить $N'_k = (N_k^2 - N_k)/2$ элементов. Мы предполагаем, что время, затраченное на вычисление одного элемента матрицы, равно константе c_1. Таким образом, время, затраченное на построение матрицы, составляет

$$T_1^k = c_1 \cdot N'_k = c_1 \cdot (N_k^2 - N_k)/2 .$$

Обратим внимание, что приведенный выше вариант затрагивает построение матрицы расстояний с нуля, то есть подходит только к первой итерации. В последующих итерациях мы можем получить матрицу расстояний, пользуясь матрицей, полученной в предыдущей итерации. Таким образом, нам необходимо пересчитать не все элементы матрицы, а лишь расстояния от измененного кластера до всех остальных, т.е.

$$N'_k = N_k$$
$$T_1^k = c_1 \cdot N'_k = c_1 \cdot N_k .$$

Заметим, что время c_1 статичным не является. При любой методике определения расстояния между кластерами при росте кластеров количество операций также возрастает. Так как количество кластеров уменьшается с

каждой итерацией, то c_1 будет также расти, поэтому правильнее будет определить его как T_p^k для k -й итерации. Таким образом имеем

$$T_1^k = T_p^k \cdot N_k' .$$

Так как размеры кластеров могут быть разными, то

$$c_1 \le T_p^k \le c_1 k^2 .$$

Левая граница соответствует одноточечным кластерам. Правая граница соответствует тому, что на k -й итерации имеем кластеры не более чем из k точек.

Шаг 2: Нахождение минимального элемента

Аналогичным образом задача решается обходом части матрицы, лежащей над главной диагональю. Время на сравнение элемента с минимальным можно считать равным константе: c_2 . Таким образом

$$T_2^k = c_2 \cdot (N_k^2 - N_k)/2 .$$

Шаг 3. Агрегирование кластеров

Этот шаг можно условно разделить на две части:

1. Предварительное агрегирование кластеров

Обход производится по соответствующим столбцам таблицы A_k и, таким образом, содержит N_k итераций. Соответственно общее время составляет

$$T_{3.1}^k = c_{3.1} \cdot N_k .$$

2. Удаление одного из кластеров

Удаление осуществляется посредством обнуления значений *k*-го столбца, а значит, снова обход по столбцу и

$$T_{3.2}^k = c_{3.2} \cdot N_k .$$

Принимая во внимание эти два подпункта, можно сказать, что $T_3^k = c_3 \cdot N_k$, где $c_3 = c_{3.1} + c_{3.2}$.

Суммарное время в последовательном случае равно

$$T = \sum (T_1^k + T_2^k + T_3^k) = \sum (T_p^k \cdot N_k + c_2 \cdot (N_k^2 - N_k)/2 + c_3 \cdot N) .$$

5.3. Оценка времени работы параллельного алгоритма для абстрактной машины

Произведем приблизительную оценку эффективности работы абстрактной машины для параллельных вычислений. Считаем, что данная машина может одновременно обрабатывать неограниченное количество потоков.

Шаг 1. Построение матрицы расстояний

Расстояния между кластерами вычисляются одновременно, следовательно, время вычисления и коэффициент ускорения будут равны:

$$t_1^k = c_1' = c_1 .$$

$$\lambda_1^k = T_1^k / t_1^k = (N_k^2 - N_k)/2 .$$

Шаг 2. Нахождение минимального элемента

Будем рассматривать наиболее очевидный способ, а именно, попарное сравнение элементов матрицы. В таком случае мы имеем

$$t_2^k = c_2' \cdot \log_2 N_k' .$$

Можно считать, что $c_2' = c_2$, в итоге это дает

$$\lambda_2^k = T_2^k / t_2^k = N_k' / \log_2 N_k' .$$

Шаг 3: Агрегирование кластеров

Произведем все замены одновременно. Таким образом мы получаем

$$t_3^k = c_3' = c_3 ,$$
$$\lambda_3^k = T_3^k / t_3^k = N_k .$$

5.4. Оценка времени работы параллельного кода на видеоадаптере

Общие положения

Обрабатываются блоки размера $n \times n$. В нашем случае $n = 16$.

Одновременно обрабатываются k блоков.

В наших расчетах обычно $k = 4$.

На самом деле общая картина выглядит несколько сложнее. Существует величина, характеризующая минимальное количество потоков $warp = 32$.

Большая часть программной реализации распараллеливания скрыта от программиста.

Наше оборудование позволяет на GPU параллельно обрабатывать $M = k \times n = 1024$ потоков.

Шаг 1. Построение матрицы расстояний

Одновременно вычисляем расстояние для 4-х блоков, поэтому

$$t_1^{k,CUDA} = c_1^{CUDA} \cdot T_1 / M \ .$$

Здесь и далее $M = k \times n$.

Пусть c_1^{CUDA} – коэффициент пропорциональности, отражающий разницу в работе центрального и графического процессора, $c_1^{CUDA} = \tau_{CPU} / \tau_{GPU}$.

Типичная ситуация $\tau_{CPU} > \tau_{GPU}$, (например $\tau_{CPU} / \tau_{GPU} = 16$), т.е. графический процессор в 16 раз медленнее центрального. Так как распараллеливание осуществляется в большее число раз, то его применение целесообразно.

Шаг 2. Нахождение минимального элемента

2.1 Нахождение минимума производим одновременно в первых 4-х блоках, затем в следующих 4-х и т.д.

Нахождение минимального элемента в блоке производим попарным мерархическим сравнением элементов. В общем случае имеем

$$t_{2.1}^{k,CUDA} = c_{2.1}^{CUDA} \cdot T_1 / M \ .$$

2.2 Формирование новой матрицы из минимумов:

Block (0,0)	Block(1,0)
-	Block(1,1)

Min(0,0)	Min(1,0)
	Min(1,1)

Матрица, возникающая на следующем шаге.

Обе матрицы формируются во внутренней памяти графического процессора.

Из k блоков выбираются минимумы. Координаты минимумов хранятся в другой матрице такой же размерности. Соответственно, имеем

$$t_{2.2}^{k,CUDA} = c_{2.2}^{CUDA} \cdot N_k / k .$$

Фактически $c_{2.2}$ является временем пересылки содержимого памяти из кэша элементарных процессоров, входящих в мультипроцессор, в разделяемую (__shared__) память, т.е. во внутреннюю память мультипроцессора.

2.3 Возвращаемся к шагу 2.1, на этот раз используя уже новую полученную матрицу. Повторяем операцию до тех пор, пока не получим матрицу размерности 1×1. Таким образом мы получим минимальный элемент матрицы. К сожалению, точную оценку произвести трудно. Грубая оценка может быть получена, если мы предположим, что работаем не с половиной, а с целой матрицей. В результате

$$N_k^1 = N_k ,$$
$$N_k^2 = N_k / n^t ,$$
$$\ldots\ldots\ldots$$
$$N_k^{t+1} = N_k / n^t .$$

Ввиду того, что мы рассматриваем только примерно половину матрицы, то можно считать, что

$$t_2^{k,CUDA} = c_2^{CUDA} \cdot (1 + 1/n + 1/n^2 + \ldots) \cdot N / 2M .$$

Шаг 3. Агрегирование кластеров

Практика показала, что этот процесс целесообразнее выполнять на центральном процессоре в виду небольших объемов вычислений и нетрудоемких операций.

Таким образом, задав предположительные теоретические «веса» c_1, удалось произвести приблизительный подсчет эффективности параллельных алгоритмов относительно последовательного.

6. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Шаг 1. Построение матрицы расстояний

Для построения матрицы расстояний с использованием CUDA алгоритм агрегирования (объединения) кластеров был вынесен в отдельный kernel-модуль, а метрика была вынесена в отдельную __device__ процедуру. Этим была достигнута возможность гибкого использования различных методов кластерного анализа за счет незначительного изменения входных данных.

Программа содержит модули, которые выполняют следующие функции.

1. Выделение памяти на GPU для исходной матрицы A и координатных данных.
2. Копирование упомянутых выше данных в GPU.
3. Выделение памяти в GPU под результат.
4. Исполнение kernel-модуля.
5. Выделение host-памяти под результат.
6. Копирование результата из памяти GPU на host.
7. Нахождение минимального элемента.

0	d10	d20	d30	d40	d50
-	0	d21	d31	d41	d51
-	-	0	d32	d42	d52
-	-	-	0	d43	d53
-	-	-	-	0	d54
-	-	-	-	-	0

Рис. 1. Разбиение элементов матрицы на блоки

При первой попытке, интеграция CUDA в исходный код поиска минимального элемента не дала особо ощутимых результатов по следующим причинам: достаточно громоздкий поиск наименьшего элемента остался реализованным без использования CUDA, и копирование результата из памяти GPU на хост оказалось слишком трудоемким. В последствии был найден оптимизированный вариант поиска минимального элемента, сводящий к минимуму нагрузку отдельных потоков мультипроцессора. Приблизи-

тельное распределение блоков и потоков на матрице расстояний можно увидеть на рис. 1. Для наглядности использовалась матрица 6×6.

Шаг 2. Нахождение минимального элемента

На этом этапе стоят следующие задачи:

1. Минимизировать процесс перемещения данных из памяти GPU на host.

2. Использовать CUDA для поиска минимального элемента матрицы.

Выполнение второй задачи осложнено одним из недостатков CUDA: общей памятью могут обладать лишь потоки внутри одного блока. Разные блоки не могут обмениваться данными. Поэтому вычисления на графическом процессоре проводились лишь частично.

1. Матрица расстояний разбивается на несколько подматриц. Каждая подматрица обрабатывается отдельным потоком.

2. В каждой подматрице находится минимальный элемент и записывается в массив в памяти GPU.

3. Массив перемещается в память хоста.

4. На хосте в массиве находится минимальный элемент стандартными средствами.

Размерность подматрицы бралась из расчета размерности блока при выполнении первого пункта алгоритма, т.е. 16×16.

К сожалению, в результате такой реализации цель достигнута не была: из-за потерь производительности при обработке циклов на GPU время поиска минимального элемента практически не было сокращено.

Был принят к рассмотрению новый рекурсивный алгоритм поиска минимума. Выглядит он следующим образом.

1. Для данной матрицы для каждого элемента выделяется один поток. Потоки объединяются в блоки 16×16.

2. Для каждого блока образуется массив размера 16×16 (он имеет вид так называемого C-файла) в разделяемой памяти (памяти, которая является общей для потоков одного блока). Таким образом, в клетку [i, j] массива записывается содержимое клетки

[bx*BLOCK_SIZE+tx][by*BLOCK_SIZE+ty] исходной матрицы.

3. В разделяемой памяти выделяется некоторая переменная vmin, которая сравнивается с элементами массива C.

4. Из значения переменной vmin для каждого блока составляем новую матрицу B (см. рис. 2).

5. Затем возвращаемся к пункту 1. Продолжаем процесс до тех пор, пока размерность матрицы B не будет равна единице.

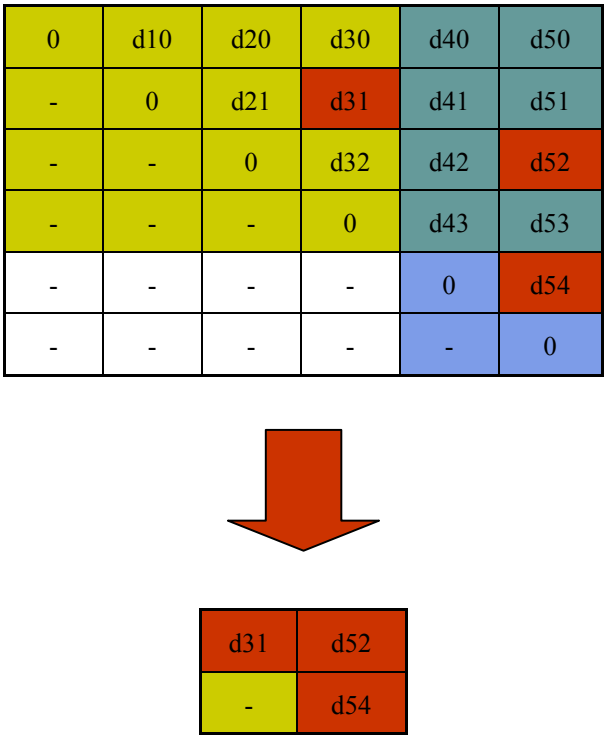


Рис. 2. Процесс конструирования следующей матрицы

Используя в первой локальной итерации исходную матрицу расстояний в качестве B, мы достаточно быстро получаем минимальный элемент в единственной ячейке матрицы, полученной в результате.

Необходимо отметить, что таким образом были достигнуты обе цели оптимизации: Матрица M больше не перемещалась из памяти устройства в память хоста, и при этом количество локальных итераций сводилось к минимуму.

Шаг 3: Копирование данных из памяти устройства в host-память.

Данная операция осуществляется стандартными методами CUDA: CudaMemCpy. Стоит заметить, что именно копирование данных является наиболее затратной по времени операции. Стоит надеяться, что с развитием аппаратной архитектуры NVidia эта проблема будет решена.

7. ЗАКЛЮЧЕНИЕ

Алгоритм иерархической кластеризации, реализованный на CUDA, позволяет проводить кластерный анализ во много раз быстрее, чем на CPU. Было проведено сравнение результатов по производительности для одних и тех же начальных данных на CUDA и CPU. Итоги можно видеть в приведенной ниже таблице и на рис. 3. Времена вычислений в таблице даны в секундах. Величина N – это количество точек, кластеризация которых производилась.

N	CPU	GPU	Коэффициент ускорения на GPU
64	0,28	0,13	2
128	1,17	0,30	4
192	3,91	0,39	10
256	10,11	0,53	19
320	21,98	0, 90	24
384	42,09	1,16	36
448	73,34	1,35	54
512	122,72	1,82	67

Сравнение результатов по производительности на CUDA и на CPU

Резюмируя, можно сказать, что были получены следующие результаты.

1. Рассмотрен алгоритм иерархической кластеризации, предложен метод отображения данного алгоритма на параллельную мультипроцессорную систему, использующуюся на современных GPU.

2. В рамках некоторых естественных предположений получены оценки времени выполнения алгоритма в последовательном случае, параллельном случае для абстрактной параллельной машины и на GPU. Также получены соответствующие коэффициенты ускорения.

3. Процедура построения матрицы расстояний между кластерами реализована на GPU средствами системы CUDA, которая при больших размерно-

стях задачи в десятки раз обгоняет по производительности ту же программу на C++ в случае ее реализации на центральном процессоре.

4. Процедура поиска минимального элемента матрицы, значительно уменьшающая обмен данными между устройством и хостом, также реализована на GPU средствами системы CUDA, и это дало большой эффект.

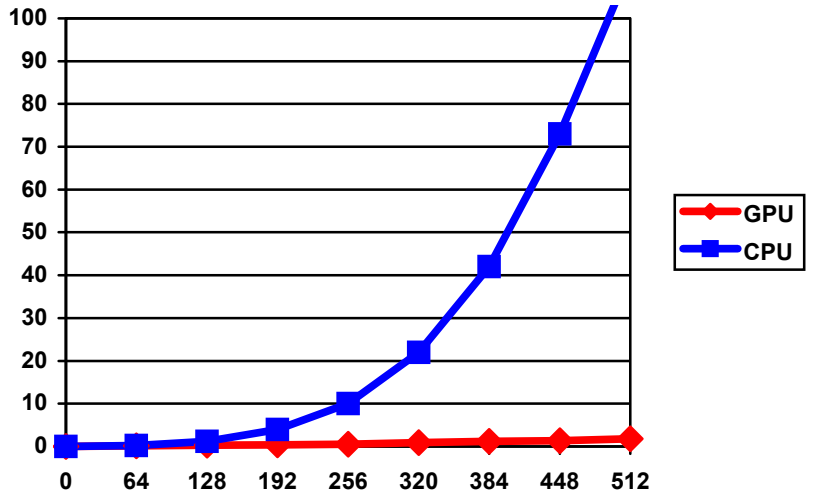


Рис. 3. Сравнительная диаграмма продолжительности работы на GPU и CPU;
по горизонтали – количество точек;
по вертикали – время в секундах

СПИСОК ЛИТЕРАТУРЫ

1. Bradley P. et al. . Scaling Clustering Algorithms to Large Databases // Proc. 4th Intl Conf. Knowledge Discovery and Data Mining. – AAAI Press, Menlo Park, Calif., 1998. – P. 8–15.
2. Zhang T. et al.. An Efficient Data Clustering Method for Very Large Databases // Proc. ACM SIGMOD Intl Conf. Management of Data. – ACM Press, 1996. – P. 103–114.
3. Huang Z. A fast clustering algorithm to claster very large categorical data sets in Data Mining // Research Issues on Data Mining and KDD, 1997. – P 367–370.

4. Ganti V. et al. CACTUS – Clustering Categorical Data Using Summaries// Proc. Int. Conf. on Knowledge Discovery and Data Mining, 1999. – P. 73-83.
5. Мурзин Ф.А. и др.. Алгоритмы определения нефтенасыщенных пластов на основе данных радиоактивного каротажа // Седьмая междунар. конф. памяти акад. А.П. Ершова, “Перспективы систем информатики”, Рабочий семинар “Наукоемкое программное обеспечение”. – Новосибирск, 2009. – С. 199–206.
6. Kalinnikov P.A. и др.. Some algorithms of image processing and their reflection onto multiprocessor systems // Joint Bull. of NCC&IIS. Ser.: Comput. Sci. – 2008. – Is. 28. – P. 67–78.
7. NVIDIA CUDA Compute Unified Device Architecture. CUDA Programming guide (v. 2.2) // NVIDIA Corporation, 2009. – 125p.
8. CUFFT Library // PG-00000-003 (v.1.1), NVIDIA Corporation, 2007. – 17 p.
9. CUBLAS Library // PG-00000-002 (v.1.0), NVIDIA Corporation, 2007. – 80 p.

Ю.Г. Платонов

ИСПОЛЬЗОВАНИЕ CQRS-ТЕХНОЛОГИИ ПРИ РАЗРАБОТКЕ КОРПОРАТИВНЫХ ПРИЛОЖЕНИЙ

ВВЕДЕНИЕ

В настоящее время в связи с широким распространением разнообразных мобильных устройств (таких как всевозможные коммуникаторы и планшетные компьютеры, использующие различные операционные системы), пользовательский рынок программного обеспечения демонстрирует спрос на приложения для корпоративных систем: складские, бухгалтерские, производственные и прочие отраслевые программы, пригодные к использованию на этих устройствах.

Для разработки таких приложений для мобильных устройств автором был предложен метод, позволяющий удаленному пользователю работать с корпоративными информационными комплексами, не имеющими web-интерфейса. Метод основан на применении архитектурного шаблона Command and Query Responsibility Segregation (CQRS) и подробно описан в [11]. Кратко суть метода применения CQRS сводится к следующему:

«*Все коммуникации системы осуществляются через систему сообщений заранее определенной структуры (contract), в результате весь процесс взаимодействия клиентской и серверной частей приложения укладывается в две цепочки событий.*

Для получения набора данных клиент отправляет публичной части сервера запрос, на основании которого из cache формируется выборка объектов, описанных в терминах клиента, после чего она возвращается клиенту как результат работы сервера.

Для попытки создать новый объект, удалить или изменить существующий объект клиент отправляет на сервер запрос-команду. Запрос проверяется на корректность и возможность исполнения и ставится в очередь. По мере достижения вершины очереди команда передается в сервис (domain model), в результате работы которого публикуется публичное сообщение. Данное сообщение сохраняется в стеке событий (event store) и при необходимости пройдя конвертацию в объект, описанный в «терминах сервиса», сохраняется в динамическом справочнике (cache).

Благодаря подобному разделению мы избегаем лишних преобразований и также можем снизить сетевой трафик».

В статье [11] не только обоснована суть CQRS-метода, но и наглядно проиллюстрировано его применение на примере инновационной разработки мобильного приложения Mobile TRIS, инициированного австралийской компанией Recruitment Systems Pty Ltd [22].

В настоящей работе автором рассматривается вопрос перспективности применения CQRS-метода при разработке произвольных стандартных корпоративных приложений.

УНИВЕРСАЛЬНОСТЬ CQRS-ТЕХНОЛОГИИ ПРИ ПРОГРАММИРОВАНИИ СТАНДАРТНЫХ ИНФОРМАЦИОННЫХ КОМПЛЕКСОВ

Разработчики мобильных клиентских приложений корпоративных программ (enterprise application) сталкиваются с необходимостью решения целого ряда нестандартных практических задач.

Например, мобильное приложение, поддерживающее работу удаленного пользователя, должно обеспечивать надежный механизм идентификации пользователей и защиты передаваемых данных [44]. Кроме того, обычно корпоративные приложения имеют сложный функционал клиентской части, который из-за ограниченности вычислительных ресурсов не может быть размещен на мобильном устройстве. При этом удаленный пользователь должен иметь возможность работать даже при отсутствии доступа к интернету, а доступ к данным в корпоративной БД может потребоваться одновременно нескольким пользователям. Часть пользователей обращаются к данным со стационарных компьютеров, остальные – с удаленных мобильных устройств.

В связи с вышеперечисленными и некоторыми другими проблемами рынок таких приложений в настоящее время остается ненасыщенным.

Рассмотрим общие аспекты организации удаленной работы с enterprise приложением. Для его обеспечения клиентская часть приложения переносится на мобильное устройство, а серверная остается неизменной.

Соответственно, при создании подобных систем особое внимание следует уделить системе взаимодействия между сервером и клиентом.

Наиболее простым и эффективным представляется перевод на мобильную платформу приложений, изначально имеющих сервисную архитектуру [33]. Клиентская (мобильная) часть приложения в этом случае представляется в виде объединения двух сервисов: стандартной программы клиента,

обеспечивающей взаимодействие с пользователем через устройства ввода-вывода и некоего хранилища последних использованных данных, и очереди команд, сохраненных клиентской частью и поставленных в очередь для исполнения на сервере, либо при активизации сервера, либо при наличии свободного места в очереди команд на сервере.

Исторически большинство функционирующих в настоящее время корпоративных приложений имеют трехуровневую архитектуру (data layer – application layer – presentation layer) [55], причем заказчик, как правило, не склонен финансировать его перевод с текущей архитектурной платформы на сервисную.

В таких ситуациях CQRS–технология может быть вполне успешно применена и при сохранении изначального трехуровневого решения.

Рассмотрим в качестве примера перспективы глубокой интеграции одного или нескольких корпоративных приложений с различными уровнями документированности API и степенью открытости кода.

Использование в этом случае CQRS–технологии позволяет объединить рассматриваемые корпоративные приложения в единую систему с помощью системы слабосвязанных сервисов. При этом, оставаясь слабосвязанной, система сохраняет свою работоспособность в случае любых изменений, вносимых в каждое из включенных в систему приложений. Приложения могут развиваться независимым друг от друга образом, в частности, расширяя свои функциональные возможности, меняя платформу и интерфейс, но соответствующие сервисы будут по-прежнему обеспечивать корректную работу всей объединенной системы.

Следует учитывать, что изначально система слабосвязанных сервисов требует некоторых трудозатрат для разработки новой архитектуры системы, но в дальнейшем, за счет возможности независимого развития каждой из подсистем, применение этой технологии представляется финансово целесообразным.

Кроме того, CQRS–технология обеспечивает безопасность данных в каждой из подсистем и четкое разграничение зон ответственности за их сохранение между системами. Поскольку подсистемы слабо связаны, непредвиденное перетекание данных между ними должно быть исключено. Пользователи каждой из подсистем, имеющие доступ к редактированию данных, отвечают за их актуальность. Пользователи других связанных подсистем могут использовать данные и результаты расчетов подсистемы только в качестве справочных данных.

В отличие от традиционной интеграции с использованием API, в системе, основанной на применении CQRS-архитектуры, может присутствовать

некий динамический справочник, основанный на данных, полученных от одного или нескольких сервисов системы и доступный только для чтения. Следует заметить, что формат данных справочника должен быть представлен в терминах сервиса, использующего эти данные.

Описанный принцип может быть успешно применен и при необходимости интеграции в одну систему стационарного и мобильного приложений.

Информационные комплексы, представленные на современном рынке, по уровню сложности интеграции с мобильным клиентским приложением могут быть отнесены к одному из основных типов, каждый из которых имеет свои особенности проектирования.

Наиболее простыми для интеграции можно считать программы с высокой степенью открытости программного кода, имеющие настраиваемые коммуникации с внешними приложениями (например, программное обеспечение компании «1С» или группа приложений, объединенных в BizTalk Server).

Платформа компании «1С» имеет специальную компоненту для обеспечения внешних соединений, работающую по принципу Com-объект и со стороны внешнего приложения, и со стороны системы «1С: Предприятие» [6].

Семейство приложений BizTalk Server представляет собой центр интеграции, который сначала преобразует сообщение в формат внутреннего представления, а после посылает его на вход преобразователя, трансформирующего сообщение в выходной XML-формат.

Трансформация осуществляется по правилам, заданным в описании преобразования, после чего полученный XML-файл преобразуется с помощью выходного преобразователя в один из заранее заданных форматов – XML, EDI или Flat file – и отправляется приложению-получателю. При разработке мобильного клиентского приложения для информационных систем этого типа следует добавить дополнительную промежуточную компоненту между мобильным клиентом и трехуровневым сервером корпоративного приложения (Рис. 1).

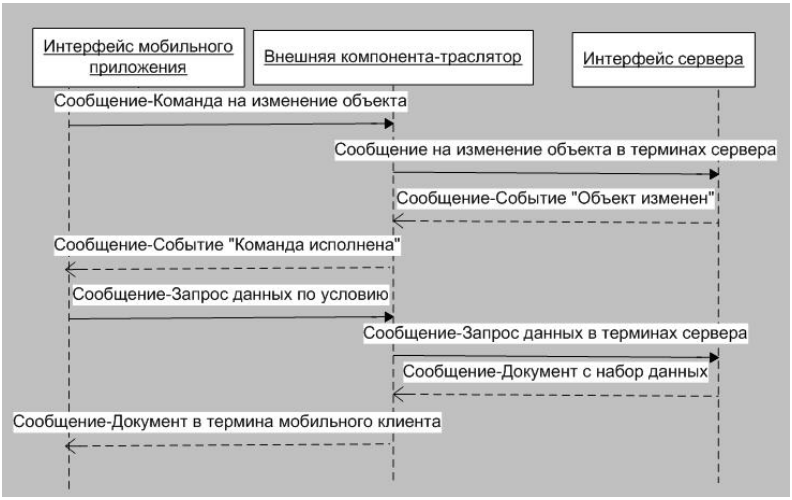


Рис. 1. Диаграмма последовательности работы промежуточной внешнего компонента – транслятора

Как видно из рисунка, этот компонент должен являться простым синхронным транслятором для трансформации сформулированных в терминах клиента сообщений-запросов во внутреннее представление сервера. Далее этот компонент должен публиковать их в заранее известный канал коммуникации. Аналогичным образом компонент обрабатывает и сообщения, публикуемые сервером.

Существует также ряд корпоративных приложений, представленных информационными комплексами без предусмотренной в них возможности коммуникации с внешними информационными системами и не имеющими соответствующего интерфейса. Они имеют закрытый программный код, но, тем не менее, их интеграция возможна и обеспечивается наличием некоторого API для доступа внешних компонентов.

Такие приложения имеют либо дополнительный компонент (например, использующий SOAP протокол), либо библиотеку с широким набором параметров для запуска с командной строки.

Для обеспечения работы мобильной клиентской части для этого типа корпоративных приложений, (как и для приложений с открытым кодом), будет создание некоторого промежуточного компонента – транслятора. Единственным принципиальным отличием является то, что в первом случае разработчик может изменять код напрямую, в то время, как во втором слу-

чае приходится использовать ограниченный набор функций программного интерфейса. Соответственно, результат зависит от его функциональных возможностей, а также полноты их документирования.

Соответственно, мы не можем быть уверены в успешном исполнении команд мобильного клиента, поэтому этот функционал должен быть перенесен в промежуточный транслятор (Рис. 2).

Следует отметить, что системы с закрытым программным кодом наиболее часто встречаются на современном рынке.

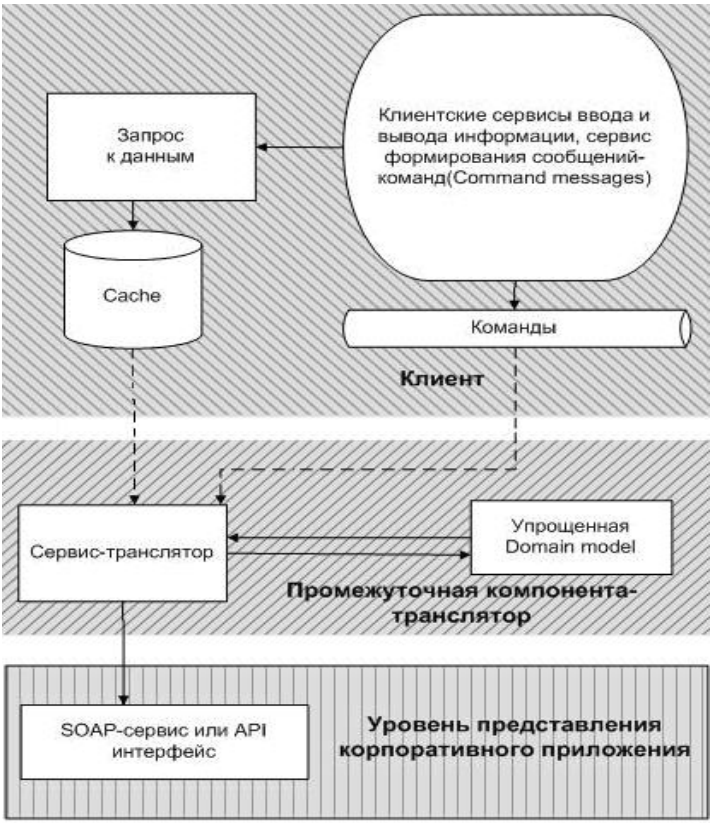


Рис. 2. Схема работы промежуточного компонента с корпоративными приложениями, имеющими собственный API

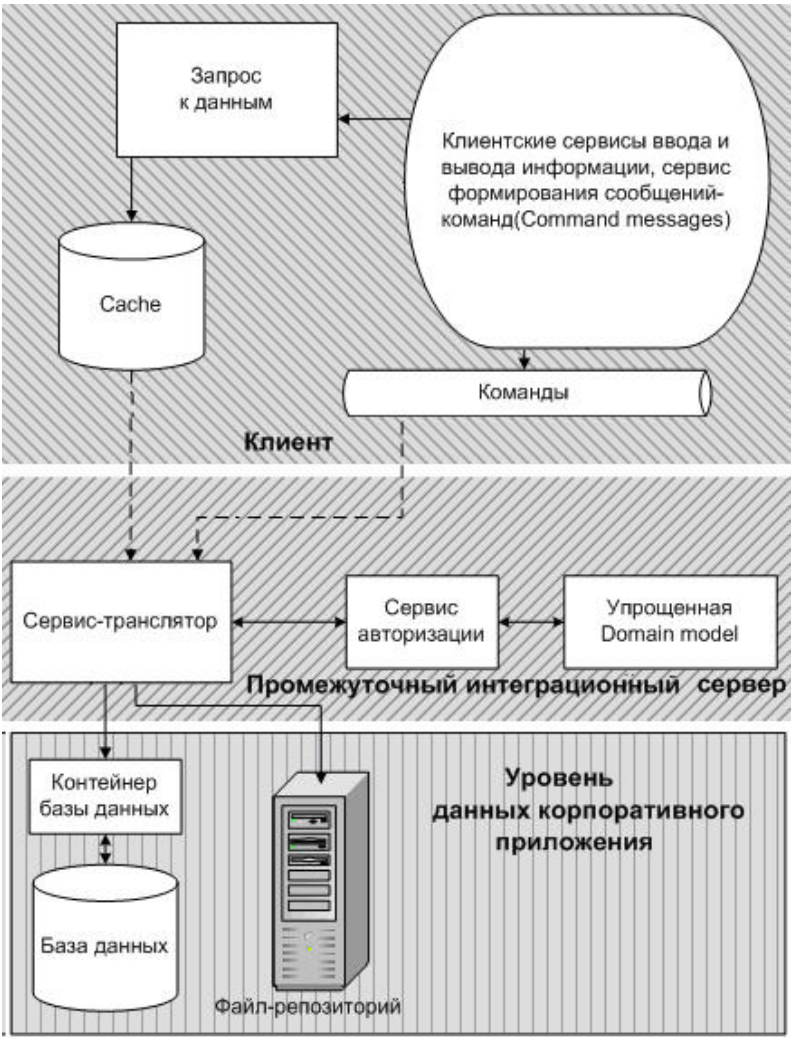


Рис. 3. Схема работы промежуточного компонента с полностью закрытыми корпоративными приложениями

Несмотря на то, что в последнее время отмечена тенденция к большей открытости программного кода и коммуникации с внешними компонента-

ми, на рынке широко представлены и корпоративные приложения с полностью закрытым программным кодом и не имеющие никакого интерфейса для коммуникации с внешними приложениями. В частности, система TRIS также относится к этому типу.

Разработка мобильной клиентской части для систем этого типа является наиболее трудоемкой.

Технология интеграции в такие системы мобильного приложения в этом случае существенно усложняется. Простая промежуточная компонента-транслятор, предложенная в первом случае, заменяется небольшим интеграционным сервером, поддерживающим связь с базой данных и файл-репозиторием. Сервер проводит собственную авторизацию пользователей и имеет некую область описания данных, которая не охватывает функционал корпоративного приложения, но логически ему не противоречит (Рис. 3).

Помимо вышеперечисленных существуют немногочисленные информационные системы, не имеющие собственного API, у которых закрыт (защищен паролем) доступ к контейнеру базы данных, или общение с файл-сервером происходит не напрямую, а по специализированному каналу. Для подобных систем создание мобильной клиентской части невозможно.

Таким образом, CQRS-метод может быть применен для стандартного информационного комплекса практически любого (кроме последнего) уровня открытости программного кода и сравнительно легко позволяет выполнить доработки, обеспечивающие возможность мобильной работы клиентской части этого приложения.

ЗАКЛЮЧЕНИЕ

CQRS-технология предусматривает различные подходы к проектированию мобильных приложений в зависимости от особенностей и уровня закрытости программного кода исходной информационной системы.

Таким образом, технология позволяет осуществлять интеграцию произвольной информационной системы с ее мобильным клиентским приложением для систем с различным уровнем сложности интеграции, что позволяет сделать вывод об универсальности метода.

СПИСОК ЛИТЕРАТУРЫ

1. Платонов Ю.Г. Разработка мобильных приложений для работы с корпоративными информационными системами // Пробл. информатики. –2011. – №3. – С. 15-32.
2. Recruitment Systems Pty Ltd. Официальный сайт, руководство пользователя: [Электрон. доку-мент]. http://ww2.recruitmentsystems.com.au/support/index.php?_m=downloads&_a=viewdownload &downloaditemid=6&nav=0,1. Проверено 03.07.11 г.
3. Фаулер М. Архитектура корпоративных программных приложений. – М.: Вильямс, 2006.
4. ГОСТ Р34.10-2001 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. Введ. 2001 г.
5. Eckerson N.,Wayne W. Three tire client/server architecture: achieving scalability, performance, and efficiency in client server applications // Open Information Systems. – 1995. – N 1. – P. 3-20.
6. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2007. – 366 с.

Н.А. Ряскина

СТРИП-ПРЕОБРАЗОВАНИЕ СИГНАЛОВ И НЕКОТОРЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

ВВЕДЕНИЕ

Важной задачей при передаче сигналов по каналам связи является уменьшение уровня помех и искажений, вносимых в различных звеньях канала, или, другими словами, повышение точности передачи сигнала по каналу. В данной работе исследовался метод повышения помехоустойчивости, известный под названием «стрип-метод» [1]. Его суть заключается в предварительном преобразовании сигнала на передающем конце путем «разрезания» его на участки равной длительности, формирования их линейных комбинаций и обратного «склеивания» в единый сигнал той же (или большей) длительности.

На приемном конце смесь сигнала с шумом, полученная из канала связи, подвергается обратной процедуре, в результате чего импульсные помехи «растягиваются» по всей длительности сигнала с одновременным уменьшением их амплитуды. Это приводит к уменьшению относительного уровня помех и, соответственно, к повышению помехоустойчивости.

Интересно, что стрип-преобразование обладает свойством, напоминающим голографический эффект. После преобразования часть сигнала может быть полностью потеряна или уничтожена. Например, значения на определенных участках приравниваются нулю. Однако при обратном преобразовании сигнал восстанавливается полностью, но с дефектами. Это происходит, потому что небольшие участки сигнала после преобразования «помнят» информацию о сигнале в целом: отсюда аналогия с голограммой.

В предлагаемой работе рассматривается стрип-преобразование сигналов, т.е. одномерных данных. Данный метод применяется и для кодирования изображений, т.е. двумерных данных. Отметим, что, хотя метод широко применяется для передачи данных со спутников (в ионосфере часть переданного сигнала или изображения может полностью пропадать), он слабо изучен на предмет того, как зависят величины погрешностей при восстановлении сигнала от величин помех в канале. В частности, в стрип-методе обычно используются матрицы Адамара (множество видов); также можно

использовать матрицы других видов. В настоящее время неясно, какие матрицы лучше подходят для каких типов сигналов.

В статье рассмотрены различные варианты стрип-преобразования сигналов на основе некоторых матриц Адамара и Хаара [2], широко применяемых в области обработки сигналов и изображений, а также в оптике [3,4]. Произведены расчеты для конкретных типов простых сигналов, определены погрешности, возникающие после восстановления сигнала. При этом моделируется импульсная помеха в канале: на преобразованном сигнале, представляющем собой вектор, в одной из позиций значение зануляется.

1. ОПИСАНИЕ ОДНОМЕРНОГО СТРИП-ПРЕОБРАЗОВАНИЯ

Первый этап стрип-преобразования одномерных сигналов состоит в представлении исходного сигнала в виде разбиения на блоки одинаковой длины.

Предполагаем, что задан вектор $\vec{v} = (v_1, \ldots, v_n)$ и $n = k \times m$.

Поэтому можно записать

$$\vec{v} = (v_1, \ldots, v_m, v_{m+1}, \ldots, v_{2m}, \ldots, v_{(k-1)m+1}, \ldots, v_n).$$

Такому вектору можно сопоставить матрицу размера $k \times m$ вида

$$A = \begin{pmatrix} v_1, & \ldots, & v_m \\ v_{m+1}, & \ldots, & v_{2m} \\ v_{(k-1)m+1}, & \ldots, & v_n \end{pmatrix}.$$

Дополнительно считаем, что задана матрица размера

$$H = \begin{pmatrix} h_{11}, \ldots, h_{1k} \\ \ldots \\ h_{k1}, \ldots, h_{kk} \end{pmatrix}.$$

Обычно в качестве H используют матрицу Адамара какого-либо вида.

Прямое стрип-преобразование состоит в том, что сначала матрицы перемножаются:

$$Q = H \bullet A = \begin{pmatrix} v'_1 & \dots & v'_m \\ & \dots & \\ v'_{(k-1)m+1} & \dots & v'_n \end{pmatrix}.$$

Здесь стоит обратить внимание на то, что любая i-я строка результата есть линейная комбинация всех исходных строк – кусков сигнала с коэффициентами из i-ой строки матрицы H. В этом и проявляется голографическое свойство преобразования: i-я строка-кусок результата содержит в себе информацию обо всех строках-кусках исходного сигнала.

Далее матрицу преобразуем в вектор и получим

$$\vec{v'} = (v'_1, \dots, v'_m, v'_{m+1}, \dots, v'_{2m}, \dots, v'_{(k-1)m+1}, \dots, v'_n).$$

Вектор $\vec{v'}$ называется результатом стрип-преобразования вектора $\vec{v}$ с помощью матрицы H и обозначается как $\vec{v'} = str_H(\vec{v})$.

Так как для матриц Адамара, Хаара и других им подобных порядка k выполнено $H \cdot H^T = H^T \cdot H = k \cdot E$, то для выполнения обратного преобразования необходимо использовать H^T. Получаем $A = \frac{1}{k} \cdot (H^T \cdot Q)$, далее по матрице A восстанавливается исходный сигнал «растягиванием» ее в строку.

Договоримся о следующих обозначениях. Будем считать, что и строки и столбцы матриц занумерованы целыми неотрицательными числами, включая ноль.

Далее, если a_{ij} – элемент матрицы A, то i – номер строки, j – номер столбца.

Наконец, импульсную помеху задаем, полагая один из элементов матрицы Q равным нулю. Очевидно, что это эквивалентно заданию нулю одного из элементов вектора $\vec{v}'$.

2. НЕКОТОРЫЕ КЛАССЫ МАТРИЦ

Матрица $A = (a_{ij})$, $(0 \le i, j \le n-1)$ называется ортогональной, если для любых i_1, i_2 справедливо $\sum_{j=0}^{n-1} a_{i_1 j} \cdot a_{i_2 j} = 0$ при $i_1 \ne i_2$, скалярное произведение любых двух различных строк равно нулю. Известно, что это эквивалентно ортогональности по столбцам. Ортогональные матрицы, элементы которых есть ±1, и называются матрицами Адамара. В качестве матрицы H из определения стрип-метода будем использовать матрицы из двух известных классов матриц Адамара и матрицы Хаара. Все необходимые понятия и определения приведены ниже.

2.1. Матрицы Адамара порядка $n = 2^k$

Матрицы порядка $n = 2^k$ определяются по индукции.
При $k = 1$ полагаем

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Если H_k уже определена, то пусть $H_{k+1} = H_1 \times H_k$, где символ $\times$ обозначает кронекерово произведение двух матриц. Если $A = (\alpha_{ij})$ есть $(n \times n)$-матрица, $B = (b_{ks})$ есть $(m \times m)$-матрица, то кронекеровым произведением $A \times B$ называется $(nm \times nm)$-матрица

$$A \times B = \begin{pmatrix} \alpha_{00} B & \alpha_{01} B & \dots & \alpha_{0n-1} B \\ \alpha_{10} B & \alpha_{11} B & \dots & \alpha_{1n-1} B \\ \dots & \dots & \dots & \dots \\ \alpha_{n-1,0} B & \alpha_{n-1,1} B & \dots & \alpha_{n-1,n-1} B \end{pmatrix}.$$

2.2. Матрицы Адамара порядка $n = p+1$, $p \equiv 3(\mathrm{mod}4)$ – простое число

Пусть p – простое число, $p \neq 2$, α – произвольное целое число, не делящееся на p. Символ Лежандра (α/p) полагается равным 1, если уравнение $x^2 \equiv \alpha(\mathrm{mod}\, p)$ имеет решение; и равным -1, в противном случае [5].

Как известно, справедлива следующая формула

$$(\alpha/p) = -1^M, \quad M = \sum_{x=1}^{p_1} \left[\frac{2\alpha x}{p}\right], \quad p_1 = \tfrac{1}{2}(p-1).$$

Здесь квадратными скобками обозначена целая часть от деления. Сразу отметим, что приведенная формула очень проста для вычислений. Конечно, при реализации на компьютере число -1 возводить в степень не нужно; достаточно контролировать четность M, например, следующим образом. Полагаем, $R = M - 2\left[\frac{M}{2}\right]$. Тогда легко видеть, что $-1^M = 1 - 2R$.

Полагаем $\chi(k) = (k/p)$ и

$$\alpha_{ij} = +1, \quad (i = 0 \text{ или } j = 0),$$
$$\alpha_{ij} = \chi(j-i), \quad (1 \le i, j \le p, i \neq j),$$
$$\alpha_{ii} = -1, \quad (1 \le i \le p).$$

В качестве примера приведем матрицу порядка 12, употребляя для краткости $\pm$ вместо ± 1, соответственно.

$$\begin{pmatrix}
+ & + & + & + & + & + & + & + & + & + & + & + \\
+ & - & + & - & + & + & + & - & - & - & + & - \\
+ & - & - & + & - & + & + & + & - & - & - & + \\
+ & + & - & - & + & - & + & + & + & - & - & - \\
+ & - & + & - & - & + & - & + & + & + & - & - \\
+ & - & - & + & - & - & + & - & + & + & + & - \\
+ & - & - & - & + & - & - & + & - & + & + & + \\
+ & + & - & - & - & + & - & - & + & - & + & + \\
+ & + & + & - & - & - & + & - & - & + & - & + \\
+ & + & + & + & - & - & - & + & - & - & + & - \\
+ & - & + & + & + & - & - & - & + & - & - & + \\
+ & + & - & + & + & + & - & - & - & + & - & -
\end{pmatrix}$$

2.3. Матрицы Хаара

Ортогональная матрицей называется квадратная матрица A с вещественными элементами, удовлетворяющая равенству $AA^T = A^T A = E$, где E – единичная матрица. Матрица Хаара – ортогональная матрица. Матрицы Хаара строятся в размерностях $n = 2^k$. Ниже приведены примеры матриц четвертого и восьмого порядков.

$$H_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

$$H_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}.$$

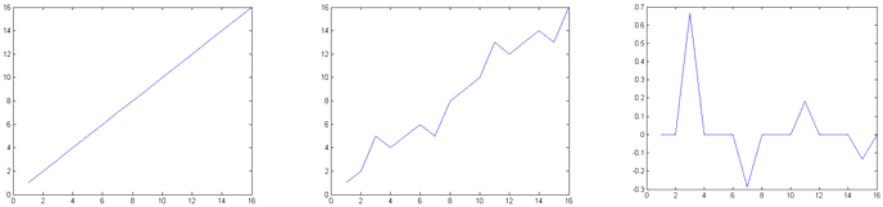
Довольно легко сформулировать общий алгоритм построения матриц Хаара. Мы видим, что при увеличении размерности последующие «волны» в два раза короче предыдущих, при этом их амплитуды умножаются на $\sqrt{2}$.

3. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

Для одномерных сигналов был проведён ряд экспериментов. В качестве исходных сигналов брались значения некоторых стандартных функций. Далее применялось стрип-преобразование, на результат воздействовали импульсным шумом, после чего производилось обратное стрип-преобразование. В качестве матриц H брали матрицы из трёх перечислен-

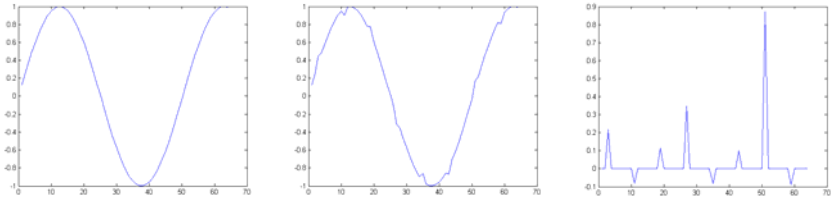
ных выше классов. В качестве шума брали $Q[2,3]=0$. В качестве погрешности бралась относительную погрешность в каждой компоненте вектора сигнала, которую вычислялась по формуле $RSD=\frac{v_i'-v_i}{v_i}$, где $\vec{v}=(v_1,\ldots,v_n)$ – точный сигнал, $\vec{v'}=(v_1',\ldots,v_n')$ – восстановленный сигнал.

Ниже приведены графики полученных сигналов и таблицы с выводом максимальных и минимальных погрешностей по модулю. При этом при подсчете минимальной погрешности не учитывается минимальная погрешность, равная нулю, то есть берется следующее значение, большее нуля.



а) б) в)

Рис.1. Стрип-преобразование линейной функции с помощью матрицы Адамара вида $n=2^k$, $n=16$:
а) исходный сигнал, б) восстановленный сигнал, в) величина погрешности (n = 16).

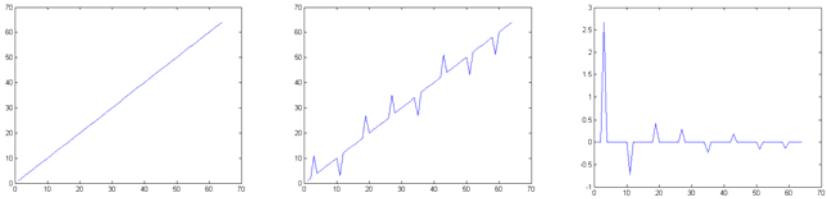


а) б) в)

Рис. 2. Стрип-преобразование функции $\sin(x/8)$ с помощью матрицы Адамара вида $n=2^k$, $n=16$:
а) исходный сигнал, б) восстановленный сигнал, в) величина погрешности

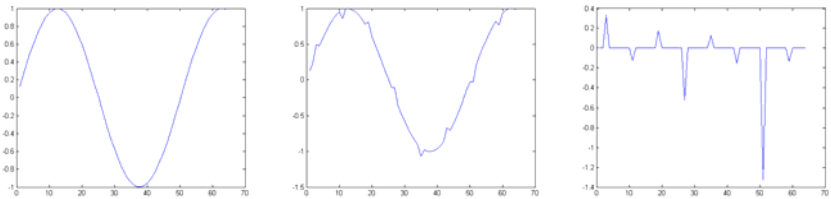
Ниже в таблице для функции $\sin(x/8)$ показаны абсолютные значения полученных величин погрешностей соответственно для различных размерностей для матриц.

	y=x		y=sin(x/8)	
	макс. погрешность	мин. погрешность	макс. погрешность	мин. погрешность
n=16 nHad=4	0.6667	0.1333	0.2556	0.0954
n=64 nHad=8	1.3333	0.0678	0.8734	0.0816
n=256 nHad=16	2.6667	0.0329	0.3433	0.0315
n=1024 nHad=32	5.3333	0.0166	0.6170	0.0620



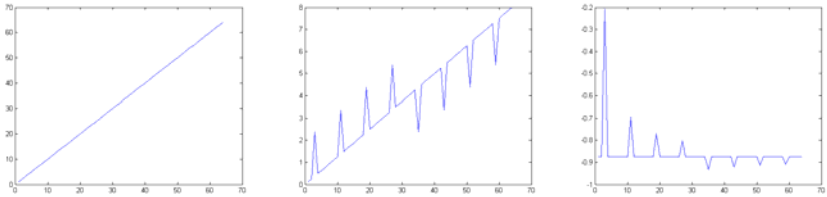
а) б) в)

Рис. 3. Стрип-преобразование линейной функции
с помощью матрицы матрицы *nLeg* = *p*+1, *p* = 3(mod4), *n* = 64:
а) исходный сигнал, б) восстановленный сигнал, в) величина погрешности



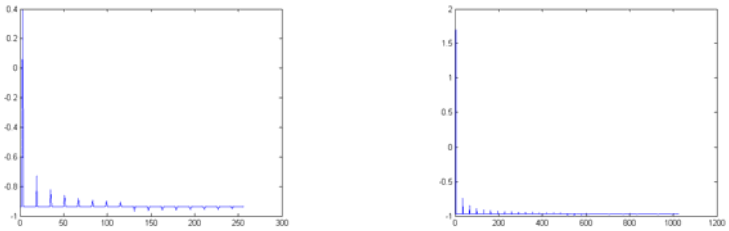
а) б) в)

Рис. 4. Стрип-преобразование функции *sin*(*x*/8), *n* = 64
с помощью матрицы матрицы *nLeg* = *p*+1, *p* = 3(mod4)
а) исходный сигнал, б) восстановленный сигнал, в) величина погрешности



а) б) в)

Рис. 5. Стрип-преобразование линейной функции
с помощью матрицы Хаара, $n = 64$:
а) исходный сигнал, б) восстановленный сигнал, в) величина погрешности



а) б)

Рис. 6. Стрип-преобразование линейной функции с помощью матрицы Хаара:
а) величина погрешности для $n = 256$, б) величина погрешности для $n = 1024$

ЗАКЛЮЧЕНИЕ

Выбор типа матрицы стрип-преобразования определяется многими факторами. В ряде случаев можно ограничиться только частью коэффициентов преобразования. Особенно это важно для сжатия массивов, несущих информацию об изображении, для его передачи, хранения и т.п. В других случаях важна величина относительной погрешности. В третьих случаях важны позиции пиков и их количество на графике относительной погрешности.

Перейдём к краткому анализу полученных результатов. Про линейную функцию можно сказать, что максимальная погрешность появляется в са-

мом начале восстановленного сигнала для всех рассмотренных матриц. С наименьшей величиной погрешности такой сигнал был восстановлен с помощью матрицы Хаара. Те же рассуждения верны и для экспоненты, но графики мы не приводим, ввиду ограниченности размера статьи. В этом случае мы имеем всего несколько пиков, один из которых по величине существенно больше остальных. Для экспоненты матрица Хаара также работает лучше. Что касается синусоиды, то здесь ситуация оказывается сложнее. Например, для синуса лучше подходит матрица порядка $n = p + 1$, $p = 3(\mathrm{mod}4)$, p – простое число. Возможно, что такого рода матрицы подходят для ограниченных периодических функций.

Полученные результаты носят предварительный, исследовательский характер и демонстрируют потенциальные возможности метода. Для получения более точных систематизированных знаний о нем необходим дальнейший, более глубокий анализ рассматриваемых вопросов.

СПИСОК ЛИТЕРАТУРЫ

1. Мироновский Л. А., Слаев В. А. Стрип-метод преобразования изображений и сигналов // Санкт-петербургский госуниверситет аэрокосмического приборостроения, 2006. – 120с.
2. Холл М. Комбинаторика. – М.: Мир, 1970. – 424с.
3. Прэтт У. Цифровая обработка изображений. Т1 – М.: Мир, 1982 – 480с.
4. Мурзина Т. Исследование светильных растровых структур на основе матриц Адамара: Дис... канд. физ.-мат. наук: 01.05.2000. – Новосибирск, 2000. – 100с.
5. Виноградов И.М. Основы теории чисел. – М.: Наука, 1965. – 172с.

С.С. Хайрулин

МОДЕЛИРОВАНИЕ БИОЛОГИЧЕСКИХ НЕЙРОННЫХ СЕТЕЙ ПРИ ПОМОЩИ ЯЗЫКА NEUROML

ВВЕДЕНИЕ

Исследования нервных систем живых организмов являются одним из наиболее развивающихся направлений биологии. Существует немало проектов целью которых являлось если не решить, то хотя бы приблизиться к пониманию работы нервных систем различных организмов. Движение в этом направлении сопряжено с основной целью нейронаук – понять, как информация распространяется и обрабатывается в нейронных контурах.

Caenorhabditis elegans, почвенная нематода, один из наиболее изученных видов живых организмов в мире, представляется идеальным кандидатом для моделирования по ряду причин. Нервная система *C. elegans* инвариантна для особей одного пола (у вида бывают мужские особи и гермафродиты) с точностью до отдельных клеток. Гермафродит является более изученным многоклеточным организмом; во взрослом состоянии его тело состоит из 959 клеток, 95 из которых – мышечные клетки, а 302 – нейроны, образующие около 5000 тысяч соединений между собой и около 2000 – между нейронами и мышцами, а также около 86 соединений с сенсорными клетками (White et al., 1986; wormatlas.org). Таким образом, *C. elegans* представляется идеальным объектом для моделирования нервной системы и может оказаться первым живым существом, чей «мозг» был воссоздан в виртуальном виде.

С момента своего появления компьютерные технологии прочно вошли во все области науки. Ученые используют компьютеры для невероятно широкого спектра задач. Для хранения больших массивов биологических данных в наше время применяются все новые и новые технологии. Для моделирования нейронных сетей было создано немало приложений[3,4]. Однако не существовало общего формального языка описания нейронов и нейронных сетей, и все вышеперечисленные программы зачастую использовали свои форматы, не поддерживающиеся другими программами. Созданный в 2004 году язык NeuroML был призван решить эту проблему [1]. NeuroML – гибкий, простой для изучения язык описания – быстро вошел в инструментарий нейробиологов и нейрокибирнетиков. NeuroML – это XML-подобный

язык, основная цель которого – задать общий формат данных для описания и обмена моделей нервных клеток и нейронных сетей. На данный момент большинство симуляторов нейронов и нейронных сетей используют формат языка NeuroML. Создание модели на языке NeuroML можно разделить на три уровня. На первом уровне описываются морфологические аспекты нейрона, начиная от сомы нейрона и заканчивая аксоном и дендритами. На втором – биофизические особенности структуры клетки, такие как строение и расположение ионных каналов, а также параметры и математические модели, симулирующие поведение канала во времени. На третьем уровне описывается расположение клетки и соединения между группами клеток.

В данной статье мы рассмотрим модель нервной системы *C. elegans*, полностью описанной на языке NeuroML, полученной из виртуальной модели червя в Калифорнийском технологическом Институте.

1. МЕТОДЫ

Как уже было упомянуто выше, модель нервной системы планируется взять из подробной модели червя, сделанной в 3D редакторе Blender (http://www.blender.org). Данная 3D модель была получена в Калифорнийском технологическом Институте (http://caltech.wormbase.org/virtualworm/). Путем кропотливого анализа огромного количества микрофотографий срезов червя была шаг за шагом восстановлена морфология *C. elegans* в виртуальной модели. Виртуальная модель представляет собой набор объектов, которые описывают соответствующие клетки живого червя, причем по морфологии объекты практически идентичны живым аналогам; каждый такой объект состоит из коллекции точек-вершин, а каждая вершина содержит в себе координаты и набор граней, которые, в свою очередь, содержат список индексов тех точек, из которых состоит грань.

В процессе работы с blender файлом появилась необходимость считывать данные из файла программно. Благодаря существующему интерфейсу и API при помощи языка python удалось импортировать необходимые данные.

К сожалению, Blender-файл содержит в себе только информацию о морфологии той или иной клетки. Отсюда следует, что существует необходимость восполнения недостающих данных о биофизических свойствах нейрона, а также о количестве и расположении синапсов между нейронами.

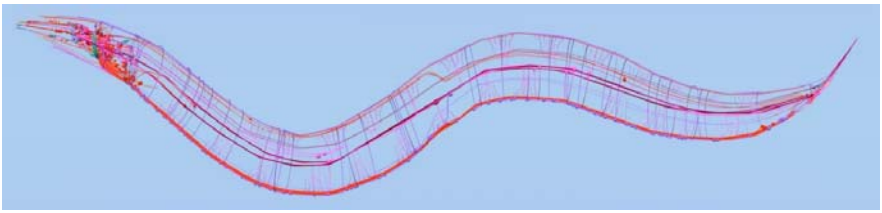


Рис. 1. На рисунке изображена модель нервной системы червя, полученной в Калифорнийском технологическом институте. Изображение получено при помощи 3D редактора Blender

1.1. Морфология нейронов

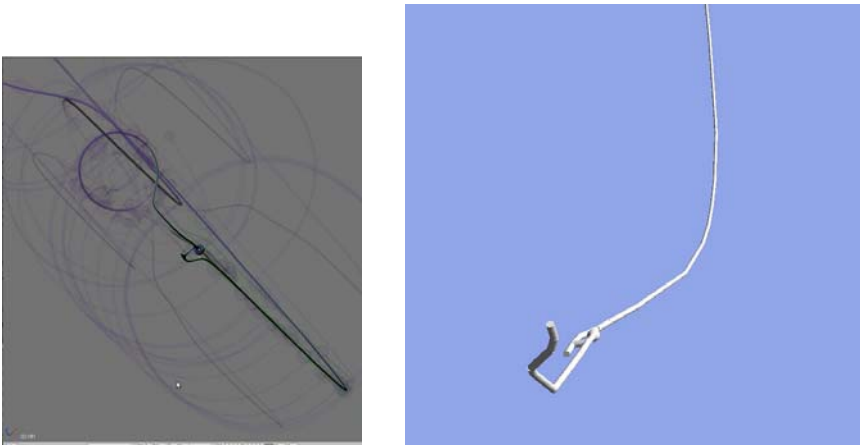


Рис. 2. На левом рисунке изображен нейрон VB1 в виртуальной модели, хранящейся в blender-файле. На правом рисунке показан тот же нейрон VB1, полученный после конвертации в NeuroML, рисунок получен в программе NeuroConstruct (http://www.neuroconstruct.org/)

Типичный нейрон может быть разделен на три основные функциональные части: дендриты, сому и аксон. Грубо говоря, дендриты играют роль “устройств ввода”, которые собирают приходящую информацию от других нейронов и передают ее далее в сому. Сома – “центральный процессор”, который выполняет важный шаг передачи. Если общий пришедший сигнал

превышает порог, генерируется выходной сигнал. Выходной сигнал передается по аксону – "устройству вывода" – другим нейронам. Морфология нейрона, как уже говорилось, была получена из Blender-файла путем трансформации данных в файле описаний нейронов в виде вершины и грани в NeuroML формат. Для решения этой задачи был написан специальный парсер, конвертирующий данные из Blender модели в NeuroML модель.

1.2. Описание биофизических свойств нейронов

Для описания модели нервной сети следует помнить и о математической модели, которая будет описывать принципы распространения сигнала по дендриту к аксону и далее от аксона к синапсу. На данный момент существует много работ, в которых довольно подробно исследуются механизмы передачи сигналов в живых нейронах. Одной из первых таких фундаментальных работ стало исследование двух биологов, Ходжкина и Хаксли [5]. Их работа была выполнена на аксоне гигантского кальмара; аксон представляет собой длинную цилиндрическую трубку, отходящую от нейрона; электрический сигнал распространяется вдоль внешней мембраны трубки. Это нервное волокно достигает толщины 0.5–1 мм (что в сотни раз превышает толщину нервных волокон млекопитающих) и представляет собой очень удобный объект для таких исследований. Помимо экспериментального исследования, А.Ходжкин и Э. Хаксли предложили модель, описывающую процессы ионного транспорта через мембрану и прохождение импульса потенциала вдоль мембраны.

Язык NeuroML позволяет также описывать биофизические свойства нейронов. В модели описываются следующие свойства нейрона: типы ионных каналов, их распределение на мембране клетки, различные электрофизические константы, такие как проводимость мембраны, сопротивление внутри клетки и т. д.; также задаются механизмы передачи сигнала. К сожалению, существующая на данный момент информация о том, какие ионные каналы есть в мембране нейронов *C. elegans,* является неполной.

1.3. Описание нейронной сети и синаптических связей между нейронами

Описание трехмерной анатомической структуры и синаптических соединений в сети свойств этих соединений также обеспечивается языком NeuroML. Таким образом могут быть определены сложные нейросети с различными типами возбуждающих и тормозящих нейронов. Есть два способа описания нейросети: явное описание положений нейронов, которое

включает в себя точное указание положений нейронов в сети, или использование алгоритмического шаблона, описывающего способ определения позиций нейронов в сети, например, случайное расположение 300 клеток в некотором трехмерном объекте. В языке NeuroML существует две основных сущности, описывающих нейросеть: популяция – определенное число клеток специфичного типа, вместе с их расположением в трехмерном пространстве; проекция определяет множество синаптических соединений между двумя популяциями нейронов или внутри одной популяции.

1.4. Структура программы

Для конвертации существующих данных из файла с виртуальной моделью были созданы специальные алгоритмы и конвертер, которые, анализируя трехмерное описание объекта в виртуальной модели, создают соответствующее описание нейрона на языке NeuroML.

1.4.1. Алгоритм восстановления морфологии нейрона из трехмерной модели

При анализе трехмерной модели было выявлено, что объекты представляют собой наборы вершин и граней; грани, в свою очередь, формируют форму нейрона. На рисунке 3 представлена схематическая трехмерная модель нейрона, состоящего из аксона и сомы.

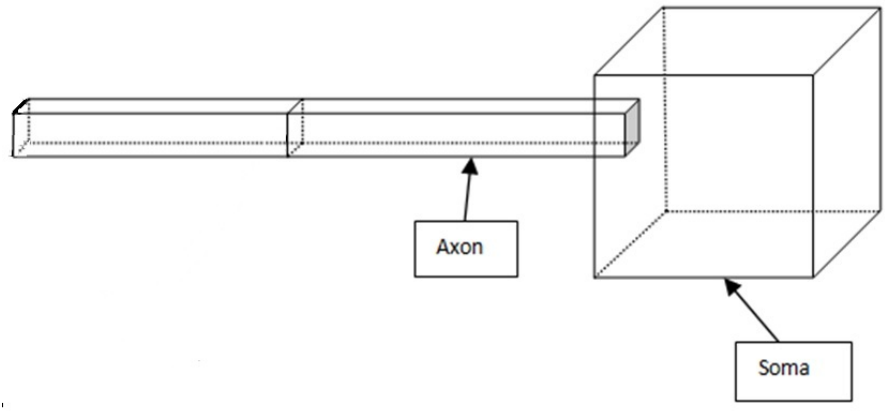


Рис.3. На рисунке представлена упрощенная трехмерная модель нейрона, содержащая сому клетки и аксон

Как можно заметить из рисунка, все части нейрона построены из сегментов “параллелепипедов” разных размеров. Единственным способом отличить отросток от сомы является высота сегмента. Любые два сегмента соединяются “срезом” (slice). Алгоритм всегда стартует с поиска сомы. Затем идет поиск всех отростков - аксонов, и дендритов. Также необходимо не забывать о ветвлениях аксона или дендритов. Ниже приведено краткое описание алгоритма и основные термины, которые используются в описании:

Vertex – вершина имеет 3 координаты (берутся из описания трехмерной модели)
Vertices – коллекция вершин
Face (грань) – это коллекция 3, 4 или более вершин. Содержат в себе индексы из массива Vertices. Также можно взять из трехмерной модели.
Faces – коллекция граней
Slice срез) – коллекция вершин, которые находятся в месте соединения двух сегментов. Не может быть гранью
Center_point – центральная точка slice. Вычисляется как центр масс slice.
Resulting_point – набор центральных точек.
Vector_len – набор вершин, отсортированных по расстоянию до center_point
Checked points collection – здесь хранятся вершины, которые алгоритм уже обработал
Adjacent point – коллекция смежных точек для заданного slice; не содержит вершин, принадлежащих checked point collection.

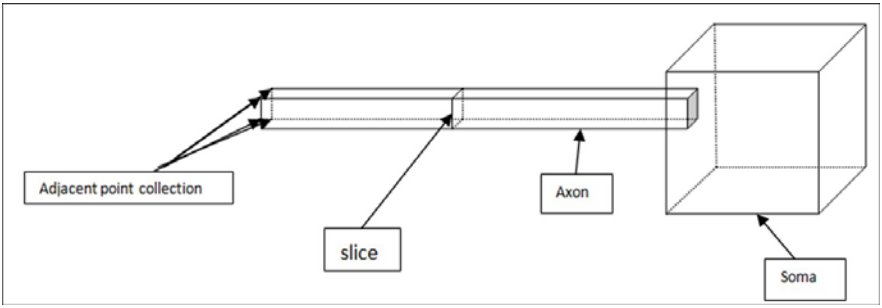


Рис. 4

Основную работу выполняет функция find_point, находящаяся в файле Entity.py.

Далее приведено описание функции find_point:

iteration = 0

После того как мы нашли сому, берется центральная точка сомы, то есть центр масс. Далее алгоритм ищет ближайший slice для текущей центральной точки.Затем

iteration += 1 center_point = центр найденного slice. Функция запускается рекурсивно с новым входным параметром center_point _slice = slice.

iteration != 0

a. If in _slice **= None** инициализируем slice из vector_len collection.
Else _slice **!=** None => slice = _slice

b. Find adjacent point for all point from slice.
Adjacent point collection содержит только те точки, которые не содержатся в checked points collection

c. Adjacent point collection может иметь 4, 6, 8 или более точек.

a. If 4 точки, то алгоритм нашел простой сегмент
без ветвлений. В этом случае мы находим slice из 4-х смежных точек, затем находим новую центральную точку для полученного slice.
Текущая center_point = новой center point, iteration += 1, кладем вершины найденного slice в checked point collection, затем заново запускаем функцию find_point с новыми аргументами.

b. If точек > 4 point.

i. If входной аргумент isBranchStart = false, то алгоритм нашел ветвление. Алгоритм находит Slice1 и Slice2 при анализе adjacent points.

ii. После определения центральных точек для найденных slice, вершины slice записываются в checked point collection, и запускается функция find_point с новыми аргументами для каждой новой center_point.

iii. If был найден один slice **And** входной аргумент isBranchStart = true

Это значит, что алгоритм нашел первый сегмент после ветвления.

В этом случае мы создаем срез из смежных точек и находим для него center_point, кладем вершины найденного slice в

checked point collection и запускаем функцию find_point с новым center_point.

iv. If не нашелся ни один slice **Or** (If нашелся 1 slice **And** isBranchStart = false)

Это значит, что мы находимся в первой точке аксона или дендрита.

В этом случае мы выбираем тот slice, периметр которого наименьший.

Считаем новый center_point, кладем вершины slice в checked point, запускаем функцию find_point с новым значением center_point

На выходе получаем набор центральных точек, который определяет ход нейрона.

1.4.2. Конвертер

Для конвертации существующих данных из файла с виртуальной моделью был создан специальный конвертер. Специальные алгоритмы, , анализируя трехмерное описание объекта в виртуальной модели, создают соответствующее описание нейрона на языке NeuroML. Конвертер был написан на языке Python. Исходные коды доступны на сайте http://code.google.com/p/openworm/. Конвертер заточен на одну определенную модель, так как его не предполагалось использовать где-либо еще, но при некоторых модификациях возможно его повторное использование в других проектах, где необходима работа с конвертацией из виртуальной модели в описательную.

2. РЕЗУЛЬТАТЫ

2.1. Модель нейронной сети *C. elegans*

К сожалению, на данный момент нет возможности проверить работоспособность модели, так как на текущей стадии проекта было создано только описание морфологической модели нейросети червя; продолжаются работы по добавлению синаптических связей и описанию механизма проведения сигнала. Разрабатываемая модель была создана в рамках более масштабного проекта OpenWorm (http://openworm.org), в контексте которого нейросеть должна стать частью модели первого виртуального организма;

она должна будет контролировать поведенческие реакции. Нейросеть должна стать управляющим элементом модели.

Результатом нашей работы стала полномасштабная модель нервной системы *C. elegans*. Наша модель включает в себя 302 нейрона, морфология каждого из которых соответствует морфологии нейрона, описанного в виртуальной модели *C. elegans*. Данные о соединениях между нейронами, а их ~7000, были взяты с сайта Wormatlas (www.wormatlas.org). К сожалению, данные о ионных каналах, обнаруженных в нейронах *C. elegans*, и о их распределении на мембранах клеток неполны.

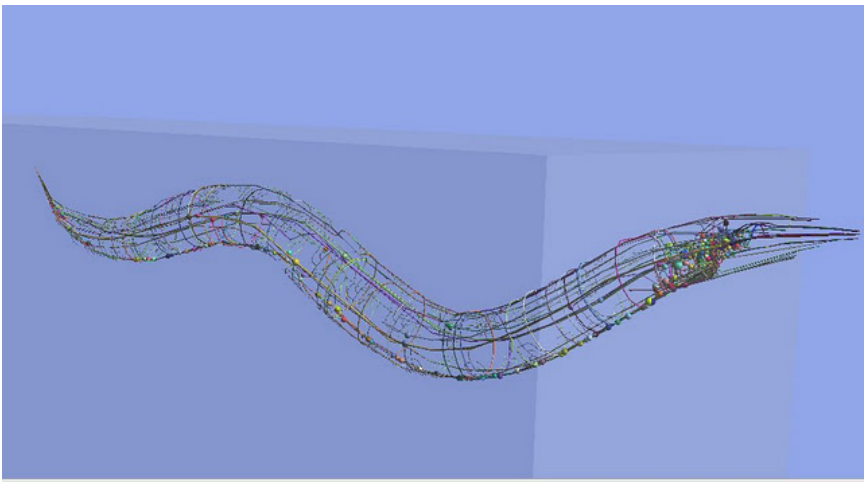


Рис. 5. На рисунке изображена модель нервной системы в NeuroML, полученной после конвертации. Рисунок получен с помощью программы NeuroConstruct

ЗАКЛЮЧЕНИЕ

Одной из основных задач современной нейроинформатики является воссоздание с большой точностью нейросети живого организма в виртуальной модели на компьютере. Предполагается, что получившаяся модель будет обладать всеми свойствами ее натурального аналога. Это, безусловно, позволит ученым из различных областей биологии и медицины развивать исследования, связанные с нервной системой. Работа над виртуальной моделью значительно упростит экспериментальную составляющую фунда-

ментальных и практических исследований. Этим обозначен огромный интерес к работе в этом направлении в наши дни. Существует немало проектов,
преследующих подобные цели, например, blueBrain project (http://bluebrain.epfl.ch/) [6]. Тем не менее о работающей виртуальной целостной нейросети еще говорить рано. В будущем наш проект должны дополнить не только другие системы *C. elegans*. Также разрабатываемая нами модель должна стать фундаментом для исследований нервных систем более сложных организмов.

СПИСОК ЛИТЕРАТУРЫ

1. Gleeson P et al. 2010 NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail. PLoS Comput Biol 6(6): e1000815. doi:10.1371/journal.pcbi.1000815
2. White J.G.et al. (1986) The structure of the nervous system of the nematode C. elegans (the mind of a worm). Philos. Trans. R. Soc. (Lond) B Biol. Sci. 314(1165):1–34
3. Hines M. et al. (2009) NEURON and Python. Frontiers in Neuroinformatics 3:1. doi:10.3389/neuro.11.001.2009.
4. neuroConstruct: A Tool for Modeling Networks of Neurons in 3D Space, Neuron, Volume 54, Issue 2, 19 April 2007, Pages 219-235.
5. Hodgkin, A.L., Huxley, A.F. (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. Journal of Physiology, 117(4): 500-544.
6. Maass, W. et al. (2002). A New Approach towards Vision Suggested by Biologically Realistic Neural Microcircuit Models. In H. Bülthoff, C. Wallraven, S.-W. Lee & T. Poggio (Eds.), Biologically Motivated Computer Vision (Vol. 2525, pp. 1-6): Springer Berlin / Heidelberg.

Т. В. Шманина

О МЕТОДЕ ВЫЯВЛЕНИЯ СИНОНИМИЧНЫХ КОНСТРУКЦИЙ ЕСТЕСТВЕННОГО ЯЗЫКА И ЕГО ПРИМЕНЕНИИ К ЗАДАЧЕ ИНФОРМАЦИОННОГО ПОИСКА

ВВЕДЕНИЕ

Ни для кого не секрет, что объем информационных ресурсов, доступных человечеству в данных момент, огромен и продолжает увеличиваться с экспоненциальной скоростью, что осложняет задачу поиска и выделения необходимой информации. Этим обусловлен большой интерес к построению различного рода поисковых систем. Дополнительной сложностью решения данной задачи является то, что подавляющая часть источников представляет собой неструктурированные или слабо структурированные тексты на естественном языке.

Над решением задачи создания «идеальной» поисковой системы, способной находить нужную пользователю информацию, трудятся уже давно. Нам известны успешные примеры реализации самых разных поисковых систем, от ИПС общего назначения, таких как Google, до узкоспециализированных систем, способных осуществлять поиск информации по коллекциям документов ограниченного круга тематик, но компенсирующих это относительно высоким качеством поиска. Однако результаты работы поисковых систем все-таки часто не устраивают пользователей в силу сильной зашумленности найденной информации (ИПС общего назначения) либо ограниченности сферы применения ИПС.

Общей тенденцией является стремление построить ИПС, учитывающую семантику запросов и способную проникнуть в семантику документов коллекции. Большинство исследователей склоняется к необходимости проведения глубокого семантического анализа текстов для создания их семантических образов, на основе использования которых можно проводить тонкое ранжирование документов в выдаче поисковой системы. Этот подход, несомненно, наиболее разумный, однако требует тщательной и долгой работы над созданием подходящих инструментов для автоматической обработки текстов. В частности, может потребоваться детальное описание каждой области знаний, всех понятий, присущих этой области, а также взаимосвя-

зей между ними. К таким способам представления семантики относятся, например, реляционно-ситуационная модель текста в поисковой системе Exactus [4], формальное графовое семантическое представление, основанное на формализме MultiNet в семантико-синтаксическом анализаторе WOCADI [5–7], RDF-триплеты, выделяемые на основе лексико-синтаксических шаблонов в поисковой системе SEUS [8].

Несомненно, перечисленные подходы имеют свои преимущества – они позволяют выделять в тестах смысловые зависимости между понятиями. Однако разработка словарей для таких систем чрезвычайно трудоемка и требует длительного времени. Кроме того, исследователи в области автоматической обработки текстов склоняются к тому, что полностью имитировать языковое поведение человека на вычислительных машинах невозможно. Поэтому имеет смысл также поиск частичных решений задачи поиска информации по смыслу.

Одним из таких частичных решений можно считать поиск тех документов, которые содержат в себе фразу, представляющую собой перефразировку запроса. В случае если заранее известны все типы запросов, которые могут быть посланы поисковой системе, а также можно предугадать, в какой форме будет встречаться в текстах необходимая информация, целесообразно пользоваться шаблонами для поиска текстовой информации, в частности, регулярными выражениями.

Однако если поисковая система не ориентирована на какую-либо одну узкую предметную область, то этот вид поиска малопригоден, так как требует указания в шаблонах конкретной лексики. В этом случае сопоставление конструкций естественного языка может основываться на анализе их синтаксических структур.

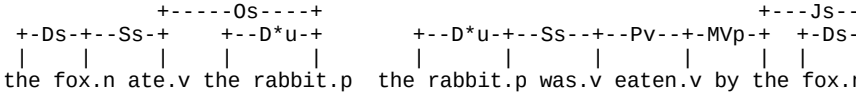
Целью данной работы являлась разработка метода, позволяющего сопоставлять конструкции естественного языка и отождествлять перефразированные варианты предложений на основе анализа их синтаксической структуры. Метод разрабатывается в предположении, что на вход подаются синтаксические диаграммы двух сравниваемых предложений, и ориентирован на использование системы синтаксического анализа предложений на английском языке Link Grammar Parser [2]. Ее особенностью является то, что получив предложение, она приписывает ему синтаксическую структуру, которая состоит из множества помеченных связей, соединяющих пары слов. Пометка каждой связи соответствует некоторому случаю правильного употребления данной пары слов в предложении и одновременно некоторому семантико-синтаксическому отношению между элементами предложе-

ния. Например, пометка S соответствует связи между субъектом и предикатом, O – между объектом и предикатом.

1. ОПИСАНИЕ МЕТОДА

1.1. Основная идея метода сопоставления естественно-языковых конструкций

Пусть даны два предложения. Для того чтобы установить, является ли второе предложение перефразировкой первого, можно сравнивать их лексический состав и синтаксическое строение, используя диаграммы связей Link Grammar Parser. Однако даже при легких перефразировках создаваемые диаграммы связей будут существенно отличаться от диаграмм исходного варианта. Это может привести к выводу, что сопоставляемые предложения различны по смыслу. Например, это наблюдается в случае двух предложений «the fox ate the rabbit» и «the rabbit was eaten by the fox»:



Поэтому предлагается провести анализ синтаксической структуры предложения на основе диаграммы Link Grammar Parser и, используя результаты этого анализа, построить новую структуру (семантический граф), которая обобщила бы синтаксис предложения и отражала основные семантико-синтаксические отношения между его смысловыми единицами. Примерами таких отношений являются связь между действием и объектом действия, действием и действующим субъектом, объектом и его свойством и т.п. В таких случаях сравнение двух предложений можно заменить проверкой на полное или частичное совпадение их семантических графов.

При этом построение семантического графа предложения проводится исключительно на основе знаний о его синтаксической структуре, а также на основе тех знаний о семантике слов, которые заложены в диаграммах связей Link Grammar Parser.

1.2. Элементы математической модели

Пусть L – множество слов естественного языка, представленного в словарях и в текстах. На L задается функция $x' = Norm(x)$, $x, x' \in L$, сопоставляющая слово с его нормальной формой. Например, произвольное существительное сопоставляется с его формой в именительном падеже и единственном числе.

Пусть POS – множество частей речи, $NF \subset L$ – множество всех слов языка, находящихся в нормальной форме. На декартовом произведении $POS \times NF$ задается двухместный предикат $PartOfSpeech(\hat{P}, x')$, который истинен тогда и только тогда, когда $x' \in NF$ и $\hat{P}$ – часть речи слова x'.

Пусть теперь $\overline{x} = \{x_1, ..., x_n\}$ – произвольное предложение над L, т.е. $\forall i\ x_i \in L$. Задается отображение $\phi : \overline{x} \to POS \times NF$, такое, что $\phi(x) = \left\langle \hat{P}, x' \right\rangle$, где $x \in \overline{x}$, $x' = Norm(x)$, и истинен предикат $PartOfSpeech(\hat{P}, x')$, сопоставляющее каждо\е слово предложения с его частью речи и нормальной формой. Это отображение однозначно, если предположить, что всегда можно однозначно определить часть речи данного слова x, например, из контекста. Пара $\left\langle \hat{P}, x' \right\rangle$ (другое обозначение – $\hat{P}[x']$) – базовое метаслово.

Кроме того, на L задаются предикаты, устанавливающие факт принадлежности данного слова $x \in L$ одной из групп вспомогательных глаголов:

1) $Vaux_{11}(x)$, где
   $x \in \{will, 'll, may, might, should, must, can, could, would, 'd, shall\}$;
2) $Vaux_{12}(x)$, где
   $x \in \{won't, shouldn't, mustn't, can't, couldn't, wouldn't\}$;
3) $Vaux_2(x)$, где $x \in \{isn't, aren't, wasn't, weren't\}$;
4) $Vaux_3(x)$, где $x \in \{hasn't, haven't, hadn't\}$;
5) $Vaux_4(x)$, где $x \in \{don't, doesn't, didn't\}$.

Связям системы Link Grammar Parser ставятся в соответствие двуместные предикаты. Если Q – название некоторой связи из множества связей

системы Link Grammar Parser, а x_1 и x_2 – два слова из предложения $\bar{x}$, то $Q(x_1, x_2)$ истинен на $\bar{x}$ тогда и только тогда, когда в диаграмме связей, построенной для данного предложения, между словами x_1 и x_2 будет существовать связь с пометкой Q. Например, для предложения «The cat chased a snake», разбор которого выглядит следующим образом:

```
              +----Os---+
  +-Ds-+---Ss--+    +-Ds-+
  |    |       |    |    |
the cat.n chased.v a snake.n,
```

будут истинны предикаты $Ds(the, cat)$, $Ss(cat, chased)$, $Ds(a, snake)$, $Os(chased, snake)$. Других предикатов рассматриваемого вида, истинных для данного предложения, нет.

Пусть предложению $\bar{x} = \{x_1, ...x_n\}$ соответствует набор базовых метаслов $\phi(\bar{x}) = \{\phi(x_1), ..., \phi(x_n)\} = \left\{\hat{P}_1[x_1^{'}], ..., \hat{P}_n[x_n^{'}]\right\}$, $P(\bar{x})$ – множество подмножеств слов из $\bar{x}$, $P(\phi(\bar{x}))$ – множество подмножеств метаслов из $\phi(\bar{x})$, а $MN = \{PredAct, PredActNo, PredPas, PredPasNo, InfAct, InfActNo, InfPas, InfPasNo, ...\}$ – множество идентификаторов типов составных слов и членов предложения (таких как инфинитивы, герундии, именные и глагольные сказуемые).

На декартовом произведении $MN \times P(\bar{x})$ задается двуместный предикат, сопоставляющий составной единице предложения идентификатор ее типа $SentenceMember(Name, U)$ – истинен тогда и только тогда, когда $U \in P(\bar{x})$ – составная единица предложения, $Name \in MN$ – идентификатор типа составной единицы предложения U.

При этом множество идентификаторов MN определяется таким образом, что

$$\forall U \left(\begin{array}{l} SentenceMember(Name, U) \rightarrow \\ \left(\neg \exists Name' \in MN \left(\left(Name \neq Name' \right) \& \, SentenceMember(Name', U) \right)\right) \end{array} \right).$$

Далее задается отображение $\psi : P(\phi(\bar{x})) \rightarrow MN \times \left(NF^{+}\right)$, такое что $\psi\left(\phi(U)\right) = \left\langle Name, \tilde{U} \right\rangle$ для любого $U = \{u_1, ...u_k\} \in P(\bar{x})$, такого, что истинен

предикат $SentenceMember(Name,U)$ и $\tilde{U} = u'_{i1}...u'_{ir}$ – конкатенация нормальных форм слов, входящих в составную единицу предложения U, $\{u'_{i1},...,u'_{ir}\} \subset \{u'_1,...,u'_k\}$. Пара $\langle Name,\tilde{U}\rangle$ (или $Name[\tilde{U}]$) – производное метаслово, соответствующее U.

Приведем принцип построения проекции отображения ψ на множество $\{PredAct\} \times NF$, где идентификатор $PredAct$ соответствует сказуемому, выраженному глаголом в активном залоге и положительной форме.

Пусть предложению $\bar{x}$ сопоставлена диаграмма связей, порожденная Link. Тогда истинность предиката $SentenceMember(PredAct,U)$ на некотором $U \subset \bar{x}$ равносильна истинности на U следующей формулы:

$$(\exists x \in U)(\exists x' \in NF)[x' = Norm(x) \,\&\, [PartOfSpeech(Verb,x') \,\&\, (\exists y \in L)\big(S(y,x)\big) \vee$$

$$(\exists y \in U)[PartOfSpeech(Verb,x') \,\&\, Vaux_{11}(y) \,\&\, I(y,x) \,\&\, \neg N(y,"not") \vee$$

$$PartOfSpeech(PartAct,x) \,\&\, Norm(y) = "be" \,\&\, Pg(y,x) \,\&\, \neg N(y,"not") \vee$$

$$PartOfSpeech(PartAct,x) \,\&\, Vaux_{11}(y) \,\&\, I(y,"be") \,\&\, Pg("be",x) \,\&\, \neg N(y,"not") \vee$$

$$PartOfSpeech(PartPas,x) \,\&\, Norm(y) = "have" \,\&\, PP(y,x) \,\&\, \neg N(y,"not") \vee$$

$$PartOfSpeech(PartPas,x) \,\&\, Vaux_{11}(y) \,\&\, I(y,"have") \,\&\, PP("have",x) \,\&\, \neg N(y,"not") \vee$$

$$PartOfSpeech(PartAct,x) \,\&\, Norm(y) = "have" \,\&\, PP(y,"be") \,\&\, Pg("be",x) \,\&\, \neg N(y,"not") \vee$$

$$PartOfSpeech(PartAct,x) \,\&\, Vaux_{11}(y) \,\&\, I(y,"have") \,\&\, PP("have","be") \,\&$$

$$\&\, Pg("be",x) \,\&\, \neg N(y,"not")]]]$$

Здесь задействовано несколько связей системы Link Grammar Parser:

I – соединяет глагол с инфинитивом;

PP – соединяет форму «have» с причастием прошедшего времени;

Pg – соединяет форму глагола «be» с причастием настоящего времени.

Формулы, аналогичные рассмотренной, можно поставить в соответствие каждому предикату $SentenceMember(Name,\cdot)$, где $Name \in MN$.

Будем считать, что для любой связи Q из системы связей Link Grammar Parser имеет место $Q(x,y) \to Q\left(\hat{P}_x[x'],\hat{P}_y[y']\right)$, где

$$x, y \in \bar{x},\ \phi(x) = \hat{P}_x[x'],\ \phi(y) = \hat{P}_y[y'].$$

Тогда каждую формулу, соответствующую предикату $SentenceMember(Name,\cdot)$, можно переписать в терминах метаслов. Полу-

ченные формулы задают порядок построения соответствующих производных метаслов $Name[\tilde{U}]$ из базовых метаслов, то есть проекцию отображения ψ на множество $\{Name\} \times NF^{+}$, $Name \in MN$. Совокупность же всех построенных проекций дает искомое отображение ψ.

Например, формула построения производного метаслова $PredAct[X]$ (здесь X представляет собой нормальную форму смыслового глагола) имеет вид:

$$V[X] \& \begin{pmatrix} (\exists N[Y])\big(S(N[Y], V[X])\big) \vee (\exists PRON[Y])\big(S(PRON[Y], V[X])\big) \vee \\ (\exists GER\_ACT[Y])\big(S(GER\_ACT[Y], V[X])\big) \end{pmatrix} \vee$$

$$I(V["Vaux_{11}"], V[X]) \& \neg N(V["Vaux_{11}"], PRTC["not"]) \vee$$

$$Pg(V["be"], PART\_ACT[X]) \& \neg N(V["be"], PRTC["not"]) \vee$$

$$I(V["Vaux_{11}"], V["be"]) \& Pg(V["be"], PART\_ACT[X]) \&$$
$$\& \neg N(V["Vaux_{11}"], PRTC["not"]) \vee$$

$$PP(V["have"], PART\_PAS[X]) \& \neg N(V["have"], PRTC["not"]) \vee$$

$$I(V["Vaux_{11}"], V["have"]) \& PP(V["have"], PART\_PAS[X]) \&$$
$$\& \neg N(V["Vaux_{11}"], PRTC["not"]) \vee$$

$$PP(V["have"], V["be"]) \& Pg(V["be"], PART\_ACT[X]) \&$$
$$\& \neg N(V["have"], PRTC["not"]) \vee$$

$$I(V["Vaux_{11}"], V["have"]) \& PP(V["have"], V["be"]) \&$$
$$\& Pg(V["be"], PART\_ACT[X]) \& \neg N(V["Vaux_{11}"], PRTC["not"])$$

.

Пусть предложению $\bar{x} = \{x_1, ..., x_n\}$ соответствуют диаграмма связей Link Grammar Parser и набор метаслов $MW = \left\{\hat{P}_{i1}[x_{i1}^{'}], ..., \hat{P}_{ik}[x_{ik}^{'}]\right\}$, состоящий из всех построенных на $\bar{x}$ производных метаслов и всех базовых метаслов, не участвовавших в построении производных, а SR – множество идентификаторов типов отношений синтаксического подчинения. Каждый идентификатор из этого множества характеризует тип синтаксического отношения между двумя значимыми словами (например, связь «сказуемое – дополнение»), а также фиксирует набор особенностей этого синтаксического отношения (например, «сказуемое в активном залоге и положительной форме, а дополнение при сказуемом – прямое»), что в целом позволяет ус-

тановить тип семантико-синтаксического отношения между двумя данными словами.

На множестве слов из $\overline{x}$ определяются трехместные предикаты отношений синтаксического подчинения $SyntacticRelation(RelationName, w_1, w_2)$ – истинен тогда и только тогда, когда w_1 и w_2 – пара значимых слов предложения, связанных отношением синтаксического подчинения типа $RelationName \in SR$, причем w_1 – главное слово в этой паре, а w_2 – зависимое.

Далее задается отображение $\chi : MW \times MW \to SR \times NF \times NF$, такое, что $\chi\left(\hat{P}_1[w_1'], \hat{P}_2[w_2']\right) = \left\langle RelationName, w_1', w_2' \right\rangle$, где $\phi(w_i) = \hat{P}_i[w_i'], w_i \in \overline{x},\ i = 1,2$, и истинен предикат $SyntacticRelation(RelationName, w_1, w_2)$. Упорядоченная тройка $\left\langle RelationName, w_1', w_2' \right\rangle$ – метасвязь, идентификатор $RelationName$ – имя метасвязи, w_1' и w_2' – соответственно главное и зависимое слова метасвязи.

Таким образом, каждая метасвязь содержит пару значимых слов, между которыми установлено отношение синтаксического подчинения. В название метасвязи при этом закладывается информация о типе распознанного синтаксического отношения, а также некоторые дополнительные характеристики, на основе которых может быть сделан вывод о том, какое семантико-синтаксическое отношение имеет место между двумя данными словами. Вся эта информация будет необходима для построения семантического графа предложения.

Отображение χ строится аналогично отображению ψ, задающему порядок построения производных метаслов. А именно, для каждого предиката $SyntacticRelation(RelationName, \cdot, \cdot)$ выписывается эквивалентная формула в терминах предикатов, соответствующих связям системы Link Grammar Parser, и предикатов $PartOfSpeech(\cdot, \cdot)$, $SentenceMember(\cdot, \cdot)$ и $SyntacticRelation(\cdot, \cdot, \cdot)$, а затем каждая эквивалентная формула переписывается в терминах метаслов и метасвязей. Полученный в итоге набор формул задает порядок построения соответствующих метасвязей из метаслов и связей Link Grammar Parser.

1.3. Семантический граф предложения

Пусть на L определен предикат, задающий отношение синонимии:

$Syn(x, y)$ – истинен, если x и y принадлежат одной части речи и являются синонимами.

Пусть $\bar{x} = \{x_1, x_2, ..., x_n\}$ – предложение над L, а $S \subset \bar{x}$ – множество всех значимых слов и составных единиц предложения из $\bar{x}$, которые, как и ранее, считаются словами.

Семантическим графом G предложения $\bar{x}$ назовем помеченный орграф, вершины которого помечены множествами синонимичных слов, а дуги – множествами семантико-синтаксических отношений:
$G = \langle V, E, s, r \rangle$, где

V – множество вершин графа G, $|V| = |S|$;

E – множество дуг графа G, $E \subseteq V \times V$;

$s : V \to P(L)$, где $P(L)$ – множество подмножеств L;

$r : E \to P(M)$, где M – множество пометок дуг.

При этом функция s должна удовлетворять условию:
$(\forall x \in S)(\exists! v \in V)\big(\big(Norm(x) \in s(v)\big) \& s(v) = \{y \in L \mid Syn(x, y)\}\big)$, то есть каждому значимому слову x предложения $\bar{x}$ будет соответствовать единственная вершина семантического графа, которая будет помечена множеством синонимов слова x.

Таким образом, пометка каждой дуги семантического графа должна выражать семантико-синтаксическое отношение между словами предложения, отобразившимися в вершины, инцидентные данной дуге.

Например, в семантическом графе предложения «Cats were sitting in the box» будет иметься дуга $sit \xrightarrow{iplaceCIR} box$, соответствующая связи между действием и обстоятельством места с оттенком «внутри».

1.4. Построение семантического графа предложения

Пусть для предложения $\bar{x} = \{x_1, ..., x_n\}$ построена диаграмма связей Link Grammar Parser. Построение семантического графа производится путем последовательного применения отображений φ, ψ и χ к диаграмме связей предложения $\bar{x}$. Так как в формулах построения метасвязей могут иметься ссылки на другие метасвязи, при реализации отображения χ распознавание синтаксических отношений необходимо производить в определенном по-

рядке. Этот порядок, в частности, задается диаграммой связей предложения, и для его определения необходимо найти в свободном дереве, полученном в результате действия отображения ψ, связь между самыми старшими членами предложения. Тогда одно из метаслов, инцидентных найденной связи, необходимо рассматривать как корень свободного дерева. Построение метасвязей в этом дереве должно производиться, начиная с листьев, таким образом, чтобы любая вершина могла образовать метасвязь тогда и только тогда, когда все вершины в поддереве, где она является корнем, уже вошли в некоторые метасвязи. Этот способ основан на том, что дочерние вершины в полученном дереве будут соответствовать зависимым от отцовской вершины членам предложения или лежать на пути к таковым. Формулы же построения метасвязей составляются таким образом, что они могут ссылаться только на те метасвязи, которые соответствуют поддеревьям с корнями в метасловах, входящих в распознаваемое отношение.

В результате применения указанных отображений получается список метасвязей, содержащих всю информацию о распознанных семантико-синтаксических отношениях на предложении. Построение семантического графа по списку метасвязей производится следующим образом. Так как в каждой метасвязи вида $MetaLinkName[w_1, w_2]$, w_1 – главное слово, а w_2 – зависимое, то в семантическом графе необходимо провести дугу от вершины с меткой w_1 к вершине с меткой w_2 и пометить ее множеством меток, соответствующих названию метасвязи $MetaLinkName$.

После того, как все метасвязи будут отображены в дуги семантического графа, каждая его вершина будет помечена множеством синонимов соответствующего ей слова. При этом множества синонимичных слов будут браться из подключенного к системе словаря синонимов. На этом построение семантического графа завершается.

Заметим, что сложность построения отображений ψ *и* χ можно оценить сверху величиной $O(Mn^{3.5})$, где M – длина соответствующего словаря формул, n – длина входного предложения. Оценка вытекает из того, что применение той или иной формулы к диаграмме связей предложения можно рассматривать как поиск изоморфного вложения дерева, заданного формулой, в дерево синтаксических связей. Для поиска же такого изоморфного вложения известны алгоритмы со сложностью, не превышающей $O(n^{2.5})$, где n – число вершин в дереве, например, алгоритм, рассмотренный в работе D. Matula [9].

1.5. Сравнение семантических графов двух предложений

Предположим, даны два предложения $\overline{x}_1$ и $\overline{x}_2$ и второе предложение необходимо сравнить с первым. Будем называть предложение $\overline{x}_1$ запросом, а предложение $\overline{x}_2$ – претендентом.

Пусть $G_1 = \langle V_1, E_1, s_1, r_1 \rangle$ и $G_2 = \langle V_2, E_2, s_2, r_2 \rangle$ – семантические графы $\overline{x}_1$ и $\overline{x}_2$ соответственно. Каждой дуге семантического графа запроса сопоставляется «равная» ей дуга в семантическом графе претендента по определенному правилу. Для этого задается отображение $F : G_1 \to G_2$ со свойствами:

1) $domF \subseteq E_1$;

2) $rangeF \subseteq E_2$ и максимальна (по весу);

3) F инъективна на своей области определения;

4) для любой компоненты связности $K = \langle V_1^K, E_1^K \rangle$ графа G_1 выполнено, что $F(E_1^K \cap domF)$ – множество дуг, также принадлежащих одной компоненте связности $S = \langle V_2^S, E_2^S \rangle$ графа G_2;

5) для любой компоненты связности $S = \langle V_2^S, E_2^S \rangle$ графа G_2 выполнено, что $F^{-1}(E_2^S \cap rangeF)$ – множество дуг, принадлежащих одной компоненте связности $K = \langle V_1^K, E_1^K \rangle$ графа G_1.

6) $\left(\forall e \in domF\right)\left(r_1(e) \cap r_2(F(e)) \neq \varnothing\right)$;

7) $\left(\forall v \in V_1\right)\left(\forall w \in V_1\right)$
$\left(\langle v, w \rangle \in domF \to \left(\left(s_1(v) \cap s_2(F(v)) \neq \varnothing\right) \& \left(s_1(w) \cap s_2(F(w)) \neq \varnothing\right)\right)\right)$

Таким образом, F ставит в соответствие две дуги графов G_1 и G_2 в том случае, если они имеют хотя бы одну общую пометку и инцидентны вершинам, помеченным синонимичными словами. Функция F с перечисленными свойствами может быть не единственной. В частности, неединственность F может быть обусловлена тем, что в G_1 или G_2 могут содержаться изоморфные подграфы. Для построения отображения F описанного вида можно использовать известные алгоритмы поиска изоморфного вложения графов, которые в общем случае имеют экспоненциальную сложность (за-

дача поиска изоморфного вложения графа является NP-полной). Однако Дж. Ульманом в 1976 году было показано, что существует алгоритм поиска изоморфного вложения, полиномиальный для каждого фиксированного подграфа [10]. Таким образом, в случае рассматриваемой задачи поиск вложения каждого фиксированного семантического графа запроса в семантический граф предложения может быть осуществлен за полиномиальное время. При этом полином будет зависеть от графа запроса.

1.5. Границы применимости метода

Предложенный автором работы метод применим только к предложениям, которые могут быть проанализированы системой Link Grammar Parser. Кроме того, метод основан на предположении, что на вход ему подается диаграмма связей, правильно отражающая все связи между понятиями. Если же она будет некорректной, то и семантический граф будет неверно отражать связи между понятиями.

Предложенный метод не может отождествлять перефразировки в том случае, если в сравниваемых предложениях содержатся формально разные системы понятий, или же понятия связаны друг с другом разными семантико-синтаксическими отношениями, например, как в предложениях «The fox attacked the rabbit» и «The rabbit fell a victim to the attack of the fox». Это происходит из-за того, что предложенный метод не учитывает семантику слов, а оперирует исключительно категориями синтаксиса.

Наконец, предложенный метод игнорирует интонации и расстановку акцентов в предложении, которые могут изменять смысл фразы. Кроме того, различные с точки зрения тонкой семантической классификации отношения могут отождествляться в случае, если их нельзя различить по грамматическим признакам.

2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Предложенный метод был реализован в рамках проекта iNetSearch аспиранта ИСИ СО РАН Александра Перфильева.

iNetSearch – это метапоисковая система, использующая лингвистические алгоритмы для извлечения документов, релевантных запросам, из множества документов, получаемых от различных поисковых систем. При этом релевантность документа определяется наличием в нем фразы поискового запроса или ее перефразировки.

Было проведено сравнение двух методов сопоставления конструкций естественного языка – оригинального (используемого в системе iNetSearch) и метода, описанного в данной статье. Оригинальный метод основан на сопоставлении диаграмм связей запроса и фразы из оцениваемого документа, причем при сравнении применяется ряд обобщений и упрощений для учета некоторых возможностей перефразирования.

Для оценки качества поиска вычислялись точность, полнота и выпадение поиска. В качестве коллекции документов рассматривалось все множество документов, полученных системой iNetSearch от поисковых систем.

По итогам тестирования было получено, что точность поиска увеличилась с 52 до 55,1%, полнота увеличилась с 87,5 до 89,3%, а выпадение снизилось с 57,6 до 50,4%. Таким образом, в среднем поисковая система стала одобрять меньше нерелевантных документов и больше релевантных.

ЗАКЛЮЧЕНИЕ

Предложенный метод отождествления перефразированных вариантов предложений на основе анализа их синтаксической структуры показал свою применимость к решению поставленной задачи и позволил улучшить работу поисковой системы iNetSearch, но, как показало тестирование, незначительно. Можно сделать вывод, что развитие предложенного метода не приведет к существенным улучшениям имеющихся результатов. Одной из причин является то, что возможности Link Grammar Parser на данном этапе работы почти полностью исчерпаны. И, несмотря на то, что Link Grammar Parser обладает рядом преимуществ (высокая скорость работы, частичный охват семантики, обилие примеров его успешного применения в системах фильтрации текстов из сети Интернет), он вынуждает оставаться на уровне синтаксиса с частичным охватом семантики. Поэтому, чтобы получить существенное продвижение, необходимо перейти на более высокий уровень, к инженерии знаний.

СПИСОК ЛИТЕРАТУРЫ

1. Батура Т.В., Мурзин Ф.А. Машинно-ориентированные логические методы отображения семантики текста на естественном языке: моногр. / Институт систем информатики им. А.П. Ершова СО РАН. – Новосибирск: Изд. НГТУ, 2008. – 248 с.
2. Temperley D., Sleator D., Lafferty J. Link Grammar Documentation [Electronic resource]. – 1998. – Mode of access: http://www.link.cs.cmu.edu/link/dict/index.html
3. Betty Schrampfer Azar. Understanding and Using English Grammar, 3rd ed. – NY: Pearson Education, 2002. – 567 p.
4. Осипов Г.С., Смирнов И.В., Тихомиров И.А. Реляционно-ситуационный метод поиска и анализа текстов и его приложения // Искусственный интеллект и принятие решений [Электрон. ресурс]. – 2008. – N 2. – С. 3–10. – Режим доступа: http://www.raai.org/library/aidt/aidt2008-2/aidt2008-2.files/2008-02_3_10.pdf
5. Sven Hartrumpf WOCADI [Electronic resource]. – Mode of access: http://pi7.fernuni-hagen.de/research/wocadi/wocadi_demo.html
6. Hermann Helbig Knowledge Representation with Multilayered Extended Semantic Networks (the MultiNet paradigm) [Electronic resource]. – Mode of access: http://pi7.fernuni-hagen.de/research/multinet/multinet_en.html
7. Hermann Helbig, Sven Hartrumpf Word Class Functions for Syntactic-Semantic Analysis [Electronic resource]. – Mode of access: http://pi7.fernuni-hagen.de/papers/helbig_hartrumpf97.pdf
8. Рабчевский Е., Крупов С., Рожков М., Булатова Г. Семантический анализ текста на основе лексико-синтаксических шаблонов для информационного поиска [Электрон. ресурс]. – 2009. – Режим доступа: http://rabchevsky.name/node/75#ir
9. D. W. Matula. Subtree isomorphism in $O(n^{5/2})$ // Ann. Discrete Math. – 1976. – N 2. – P. 91–106.
10. Julian R. Ullmann. An algorithm for subgraph isomorphism // Journal of the ACM 23 (1). – 1976. – P. 31–42.

Т. В. Шманина

ОБЗОР МЕТОДОВ ПРЕДСТАВЛЕНИЯ СЕМАНТИКИ ТЕКСТОВ И ИЗВЛЕЧЕНИЯ ЗНАНИЙ ИЗ НИХ

ВВЕДЕНИЕ

В условиях стремительного роста объема информационных ресурсов все более актуальной становится задача автоматического извлечения знаний из текстов на естественном языке.

Данная статья посвящена обзору некоторых методов формального представления знаний и семантики текстов в целом, а также способов извлечения знаний из неструктурированных текстов.

1. «СИЛЬНЫЕ» МЕТОДЫ ПРЕДСТАВЛЕНИЯ СЕМАНТИКИ

Методы этого типа изначально возникли в рамках работ по машинному переводу. Они характеризуются использованием семантических метаязыков и специальных языков для описания значений предложений. Примером такого подхода служит работа Мельчука, кратко описанная в [1]. В рамках этого подхода язык рассматривается как очень большая модель, в которой определены лексические предикаты (функции), например, такие, как *Caus*, *Lab*, *Perf* и т.д. (см. [1]), а каждое значение слова должно быть описано семантической формулой, задающей его семантические валентности. Множество всех таких описаний представляет собой семантический (толково-комбинаторный) словарь языка. Набор статей в толково-комбинаторном словаре можно считать некоторой подмоделью исходной модели, являющейся языком.

Статья толково-комбинаторного словаря несет информацию о валентностях конкретного слова, верную не только в ее рамках, но и в рамках всего языка в целом. Валентности соответствует предикат $P^{val}(c_x, \overline{y})$, где $\overline{y} = y_1, ..., y_n$ – семантические актанты слова c_x, n – валентность слова c_x. Например, в предложении *«Петя читает книгу»* будет $c_x = "читать"$,

$n = 2$: $y_1 = "Петя"$, $y_2 = "книга"$, т.е. условно можно написать $P^{val}(c_x, y_1, y_2) = 1$.

Таким образом, значения отдельных слов и целых предложений могут быть выражены специальными формулами. Например, одно из значений слова *потушить* описывается формулой *Perf Caus(im,Fin Lab(vin,ОГОНЬ))*, где *огонь* – это базисное, далее не разложимое понятие, а *Perf*, *Caus*, *Fin*, *Lab* – это базисные семантические (лексические) функции. Так, базисная функция *Lab(x,y)* обозначает, что аргумент, обозначаемый словом *x,* подвергается действию аргумента, обозначаемого словом *y*.

Значение предложения описывается математическим выражением. Например, предложению «*Пожарники потушили загоревшийся сарай»* соответствует выражение *Perf Caus (ПОЖАРНИКИ,Fin Lab(Perf Incep Lab(САРАЙ,ОГОНЬ),ОГОНЬ))*, которое в обратном переводе на русский язык имеет следующее значение: «*Пожарники сделали так, что перестал подвергаться действию огня начавший подвергаться действию огня сарай»*. Таким образом, значение предложения формально можно представить единственным образом.

Так как описанный метод и прочие «сильные» подходы были изначально созданы для целей автоматического перевода текста с одного языка на другой, они вынужденно претендуют как на полноту охвата множества интерпретируемых выражений естественного языка, так и на точность их семантического описания. Однако из-за большой сложности и трудоемкости реализации таких подходов ни одна из попыток их применения и по сей день не увенчалась успехом.

2. «СЛАБЫЕ» МЕТОДЫ ПРЕДСТАВЛЕНИЯ СЕМАНТИКИ

В настоящее время многие исследователи, работающие в области представления семантики текстов, склоняются к необходимости выделения в текстах семантически значимых объектов и нахождению семантических или семантико-синтаксических связей между ними. В итоге текст представляется в виде набора отдельных «троек», состоящих из объекта действия, действующего субъекта и связи определенной семантической категории между ними, либо в виде семантических сетей, либо набора фреймов. К интересным работам такого типа можно отнести следующие.

2.1. Фреймовая модель в RCO Fact Extractor

Подход, предложенный разработчиками RCO [2] (Киселев С.Л., Ермаков А.Е., Плешко В.В.), главным образом ориентирован на анализ синтаксической структуры предложений и основан на применении специальных шаблонов к сети синтактико-семантических отношений между словами текста. Эти шаблоны позволяют распознавать и интерпретировать допустимые способы описания ситуации в тексте и определяют способ интерпретации фрагментов сети в заданные фреймы с идентификацией участников ситуаций и их ролей. Ориентированность шаблонов на применение не к тексту, а к сети синтактико-семантических отношений между словами обеспечивает их высокую инвариантность к особенностям поверхностно-синтаксической организации предложений.

Таким образом, анализ семантики текста разбивается на следующие этапы.

Построение сети синтактико-семантических отношений

В результате синтаксического анализа предложения и последующих трансформаций дерева синтаксических зависимостей между словами формируется сеть синтактико-семантических отношений. Семантическая сеть содержит все сущности, упоминавшиеся в тексте предложения – наименования предметов и лиц, действий и признаков, связанные различными типами синтактико-семантических связей. Направление связи обычно соответствует направлению синтаксического подчинения слов. Пример семантической сети представлен на рисунке 1.

Узлы и связи в сети имеют набор следующих основных атрибутов:

1. Часть речи слова, соответствующего узлу.
2. Семантический разряд референта узла, например: *персона*, *организация* и др.
3. Строка текста, соответствующего узлу, в нормальной форме.
4. Тип синтактико-семантической связи между узлами, например: *аргумент (совершить -> сделку), принадлежность (акция -> Геркулеса).*
5. Семантический падеж и коннектор – предлог или союз, при помощи которого устанавливается связь (альтернатива традиционным семантическим ролям (субъект, объект, инструмент, локатив и т.п.)).

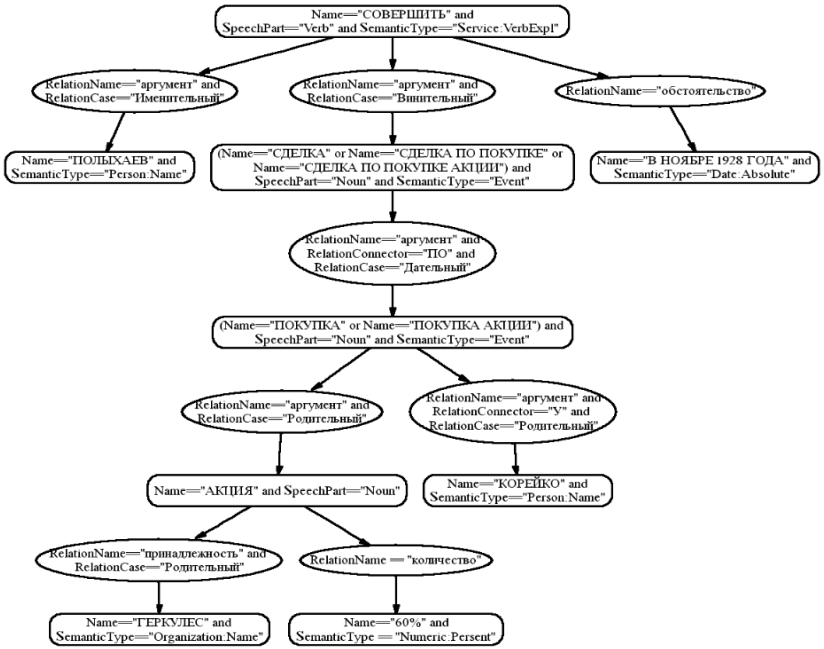


Рис. 1. Пример семантической сети, соответствующей предложению «В ноябре 1928 года Полыхаев совершил сделку по покупке 60% акций ООО “Геркулес” у Корейко»

Семантическая сеть инвариантна к синтаксической структуре и порядку слов с точностью до структуры пропозиции, выбранной автором для описания ситуации.

Фреймы и семантические шаблоны

Логическая схема ситуации в терминологии искусственного интеллекта называется фреймом. Фрейм имеет имя, которое идентифицирует тип описываемых им ситуаций (например, купля-продажа акций), а также содержит слоты, которые имеют свои имена, идентифицирующие роли участников ситуации.

Для семантической интерпретации каждого способа описания ситуации в тексте используется соответствующий синтактико-семантический шаблон, задающийся в виде сети, изоморфной искомой в тексте, в узлах и свя-

зях которой при помощи логических выражений указываются условия, которым должны удовлетворять узлы и связи искомой сети. В рамках описываемой модели семантическая интерпретация описания ситуации в тексте есть поиск в его семантической сети такой подсети, которая изоморфна шаблону, с заполнением слотов соответствующего фрейма именами участников ситуации из текста в соответствии с ролями, указанными в узлах шаблона.

Приведенный метод, с точки зрения автора статьи, обладает рядом преимуществ, в числе которых стоит отметить лингвистическую прозрачность метода, упрощающую поддержку и разработку реализующей его системы извлечения фактов, а также высокую точность работы метода на подходящих входных данных, то есть при условии построения корректной синтактико-семантической сети предложения и наличии шаблонов фреймов, с помощью которых можно распознать семантические отношения в данном предложении. Однако стоит отметить, что правильность результатов работы предложенного метода напрямую зависит от правильности построения синтаксического дерева предложения. Задача же построения синтаксического разбора является чрезвычайно сложной, и большинство существующих на данный момент систем работают недостаточно удовлетворительно, в особенности на обычных, зашумленных текстах. Также для каждого распознаваемого системой отношения требуется создание шаблонов для всех возможных вариантов его описания в текстах (с точностью до вариаций, соответствующих одной пропозиции), так как неописанные языковые явления не будут распознаваться системой. Создание же исчерпывающей базы шаблонов является чрезвычайно трудоемкой задачей, что делает метод более подходящим для создания систем извлечения фактов специального назначения.

2.2. Фреймовая модель Симакова представления знаний.

В модели текста и извлечения знаний, предложенной К.В. Симаковым и др. [3, 4], текст рассматривается в виде последовательностей сегментов. Минимальными элементами сегмента являются слова и знаки препинания.

Данная модель текста представима в виде алгебраической системы вида $TM = \langle T, W, t_{\text{O}}, \bullet \rangle$, где T – множество текстовых сегментов, W – множество слов, t_{O} – пустой текстовый сегмент, $\bullet$ – операция сцепления на T. В модели текста определены следующие свойства:

1. $\forall w \in W \Rightarrow w \in T$ – каждое слово является текстовым сегментом.

2. $(\forall t_1 \in T)(\forall t_2 \in T)(\exists! t = t_1 \bullet t_2 \,\&\, t \in T)$ – операция сцепления позволяет из произвольной пары текстовых сегментов сформировать новый текстовый сегмент.

3. $(t_O \in T) \& (\forall t \in T \Rightarrow (t = t_O \bullet t) \& (t = t \bullet t_O))$ – пустой текстовый сегмент является нейтральным элементом по отношению к операции сцепления.

4. $((t_1, t_2 \in T) \& (t_1 \neq t_O) \& (t_2 \neq t_O)) \Rightarrow (t_1 \bullet t_2 \neq t_2 \bullet t_1)$ – некоммутативность операции сцепления.

Из этих свойств следует:

1. Любой текстовый сегмент может быть представлен в виде сцепления слов.

2. Поскольку слова являются неделимыми сегментами, то удобно измерять длину сегментов в словах; далее длина сегмента *t* обозначается N_t.

3. Длина пустого сегмента t_O равна нулю.

В качестве модели представления знаний используются фреймы. Фреймовая модель задается пятеркой $FA = \langle F, S, T, R_{FS}, R_{ST} \rangle$, где F – множество фреймов; S – множество фреймовых слотов; T – множество значений слотов; $R_{FS} \subseteq F \times S$ – отношение, задающее связи между слотами и фреймами; $R_{ST} \subseteq S \times 2^T$ – отношение, задающее для каждого слота допустимую область значений.

В данной работе полагается, что *FA* задается человеком-экспертом, который определяет все возможные фреймы и составляющие их слоты. Также полагается, что все возможные значения слотов *T* представимы в виде текстовых сегментов модели *TM*.

Модель извлечения

Ключевыми компонентами модели является множество правил извлечения *V*, множество образцов *P* и множество элементы образцов *R*. Любой образец может быть представлен в виде сцепления *n* элементов – $\forall p \in P \Rightarrow p = r_1 \bullet ... \bullet r_n$. Любое правило извлечения представляется в виде сцепления трех образцов: префиксного, извлекающего и постфиксного – $\forall v \in V \Rightarrow v = p_b \bullet p_c \bullet p_a$. Префиксный и постфиксный образцы могут быть пустыми (т.е. нейтральными по отношению к операции сцепления). В модели извлечения введена функция покрытия $a : T \times V \rightarrow \{$*истина*, *ложь*$\}$. Данная функция для любого правила извлечения и любого текстового сег-

мента позволяет ответить на вопрос, покрывает ли данное правило данный текстовый сегмент. Функция покрытия также применима для образцов и их элементов. Правило $v = p_b \bullet p_c \bullet p_a$ покрывает текстовый сегмент, если этот сегмент представим в виде тройки $t_b \bullet t_c \bullet t_a$, и каждый из этих сегментов покрывается соответствующим образцом из тройки $p_b \bullet p_c \bullet p_a$. Образец $p = r_1 \bullet ... \bullet r_n$ покрывает текстовый сегмент, если этот сегмент представим в виде $t_1 \bullet ... \bullet t_n$, и каждый t_i покрывается соответствующим r_i. Функция покрытия для элемента образца определяется внутренней структурой элемента. Если правило покрывает текстовый сегмент, то извлечению подлежит та часть текстового сегмента, которая покрывается извлекающим образцом правила.

Отсюда следует связь между моделью извлечения и моделью фреймов:

1.

$$\left(\forall s \in FA\right)\left(\exists V_s \subset V\right):\left(\forall v \in V_s\right)\&\left(\forall t = t_b \bullet t_c \bullet t_a \in T\right)\&\left(a(t,v) = \textit{истина}\right) \Rightarrow$$
$$\Rightarrow \left(t_c \in T_i : sR_{ST}T_i\right)$$

с каждым слотом *s* связан набор правил *s V,* такой, что любой текстовый сегмент, извлекаемым одним из правил из *s V*, принадлежит области значений данного слота.

2. $\left(\forall s_1, s_2 \in FA\right)\left(\exists V_{s_1}, V_{s_2} \subset V\right): V_{s_1} \cap V_{s_2} = \mathrm{O}$: множества правил извлечения для каждого слота уникальны и не пересекаются между собой.

Чтобы дать интерпретацию функции покрытия для элементов образцов, рассмотрим структуру элемента $r_i = \langle c, e, l_1, l_2 \rangle$ где $c \subseteq W$ – лексическое ограничение, $e \subset W$ – исключение лексического ограничения, l_1 и l_2 – минимальная и максимальная длина покрытия элемента. Лексическое ограничение *c* и его исключение *e* определяют множество слов *c\e={w}*, которые могут встречаться в текстовых сегментах T_{ri}={*t*}, покрываемых элементом r_i. Слова {*w*} берутся из множества *W* модели текста. Минимальная и максимальная длины покрытия l_1 и l_2 определяют допустимый диапазон длин текстовых сегментов T_{ri}. Таким образом, чтобы элемент *r* покрывал текстовый сегмент *t*, необходимо, чтобы все слова, сцепление которых образует *t*, принадлежали множеству слов, разрешенных лексическим ограничением элемента, не попадали в исключения, а длина текстового сегмента должна находиться в диапазоне [l_1,l_2].

Заметим, что данная модель является обучаемой, что является ее несомненным достоинством, так как в значительной степени упрощает реализацию основанных на ней систем извлечения фактов. Однако, как и в случае

всех обучаемых систем, при неудачно подобранной или недостаточной по объему обучающей выборке в модели либо может появиться большое число некорректных правил извлечения, либо может быть получено множество правил, покрывающее лишь немногие явления языка, что бывает сложно проконтролировать после завершения этапа обучения.

2.3. Семантические сети в WOCADI Parser

WOCADI [5, 6, 7] – семантико-синтаксический анализатор, разработанный в университете в Хагене, Германия. Данный анализатор преобразует текст на немецком языке в формальное семантическое представление.

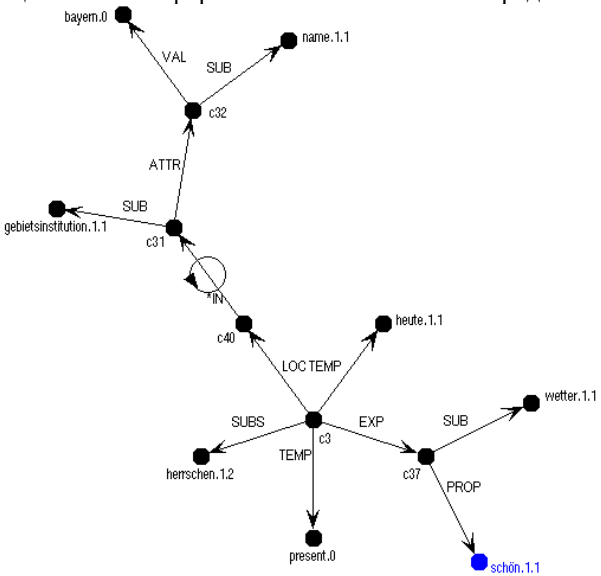


Рис. 2. Семантическая сеть предложения

Анализ текста системой WOCADI проводится в несколько этапов:

1. *Графематический анализ*: производится разбиения текста на предложения и слова.
2. *Морфологический и лексический анализ текста.* При этом используются несколько словарей: HaGenLex (содержит детальную морфосинтаксическую и семантическую информацию о словах), семантически неклассифицированный лексикон и словари имен собственных. Морфо-лексический анализатор определяет нормальные формы

слов, извлекает информацию об их грамматических формах, заложенную в суффиксах, префиксах и инфиксах, и возвращает большой перечень выделенных признаков (от 20 до 80 признаков) для каждого слова. Также производится анализ структуры и семантики составных слов, обилие которых характерно для немецкого языка.

3. *Семантико-синтаксический анализ текста*: производится построение связей между словами предложения на основе использования функций классов слов (WCF). Результатом работы семантико-синтаксического анализатора являются семантические сети.

Каждая такая семантическая сеть состоит из элементов двух основных типов: понятий, таких как «компьютер» или «персик», и отношений между этими понятиями, например, «персик является фруктом». На рис. 2 приведен пример такой сети для предложения «In Bayern herrscht heute schönes Wetter» («В Баварии сегодня хорошая погода»).

Разработчиками данного анализатора было показано, что WOCADI в совокупности с базой знаний может применяться для перевода запросов, сформулированных на немецком языке, на стандартные языки запросов, в частности SQL.

По мнению автора статьи, главным достоинством описанного метода является возможность достаточно глубокого семантического анализа предложения, результатом которого являются сети отношений, удобные для дальнейшей обработки. Также следует отметить, что в силу особенностей алгоритма анализа синтаксической структуры предложения данная система является устойчивой к некорректным входным данным, таким как грамматически неправильно построенные фразы. Однако одновременно с этим качество работы подобной системы критически зависит от полноты семантических словарей, которые создаются и корректируются вручную опытными специалистами, что делает процесс разработки таких систем чрезвычайно трудоемким.

2.4. Реляционно-ситуационная модель текста в ИПС Exactus

Exactus [8] – система семантического поиска и анализа текстовой информации. Общая схема анализа предложения, использующаяся в этой системе, выглядит следующим образом:

1. *Морфологический анализ*: в тексте распознаются слова и разделители, для каждого слова определяется список всех возможных грамматических форм, формируется множество предложений, каждое из ко-

торых содержит упорядоченный список слов, а каждому слову сопоставлено несколько вариантов омонимичных лексем.

2. *Синтаксический анализ*: происходит установление различных зависимостей между лексемами и выделение синтаксем (минимальных синтактико-семантических единиц языка, несущих обобщенный категориальный смысл и характеризующихся взаимодействием морфологических, семантических и функциональных признаков). Выход анализа – предложение в виде списка деревьев синтаксического подчинения (на основании эвристических правил выбирается наиболее вероятный вариант синтаксического разбора) и множество синтаксем.
3. *Реляционно-ситуационный анализ*: выявление значений синтаксем (например, каузатив, результатив, адресат и т.д.) и семантических связей между ними (например, «CAUS(каузатив, результатив)»), которые будут образовывать семантический образ документа. Для этого сначала определяется предикатное слово (глагол, предикативное наречие и т.п.). Затем устанавливаются значения синтаксем и отношений на них. Значения синтаксем, связанных с данным предикатным словом, определяются по лингвистическим словарям, содержащим сведения о синтаксической сочетаемости каждого глагола с синтаксемами и указания на то, как могут быть связаны между собой именные синтаксемы. После того, как установлены значения синтаксем при предикатном слове, определяются устойчивые отношения между ними: в статье словаря, соответствующей рассматриваемому предикатному слову, ищутся пары значений синтаксем, образующие элемент семантического отношения. В безглагольных предложениях значение синтаксемы определяется по правилу, посылка которого содержит характеристики контекста синтаксемы, а следствие указывает значение, которое необходимо установить рассматриваемой синтаксеме.

Результатом работы описанных процедур анализа являются структуры, описывающие семантическую информацию, передаваемую текстом, в виде неоднородных семантических сетей. Эти семантические образы текстов запросов и документов используются в вычислении степени семантической близости запроса и документов при поиске и ранжировании.

Например, если имеется вопросительный запрос «К чему приводит окисление наночастиц?», то ответом может быть предложение «Окисление наночастиц приводит к формированию структуры наночастицы – металлическое ядро – оксидная оболочка», т.к. именная группа «к формированию

структуры наночастицы – металлическое ядро-оксидная оболочка» имеет семантическое значение «результатив», как и синтаксема «к чему» в запросе. На рис. 3 приводятся семантические сети для запроса и ответа.

К достоинствам данного метода следует отнести возможность его применения в вопросно-ответных системах, а также высокую точность распознавания семантических отношений вследствие использования словарей, содержащих детальную информацию о семантике слов. В то же время, как и в предыдущем случае, качество семантического разбора текста в значительной степени здесь будет зависеть от полноты и качества семантических словарей. Так как такие словари должны обязательно корректироваться специалистами в области семантики вручную, такие системы характеризуются сложностью построения и поддержки.

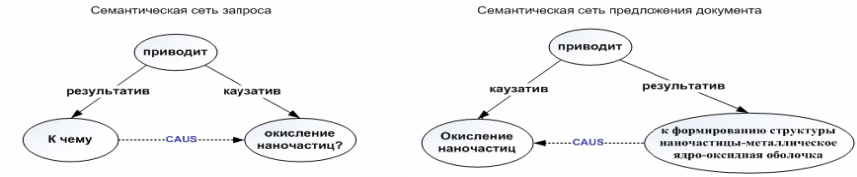


Рис. 3. Семантические сети запроса и предложения, являющегося ответом

2.5. Лексико-синтаксические шаблоны в поисковой системе SEUS

SEUS [9] – проект поисковой системы, основанный на автоматическом построении семантического представления текста в виде семантической сети. В данной поисковой системе поиск соответствующих запросу документов производится по элементам построенной семантической сети (RDF-триплетам).

Триплеты предлагается извлекать из текста при помощи лексико-синтаксических шаблонов – характерных конструкций (т.е. словосочетаний и оборотов) из соответствующих элементов языка. Такой метод использует иерархию шаблонов, которые состоят главным образом из индикаторов части речи и групповых символов.

Тело шаблона состоит из входной и выходной схем. Входная схема – характерное описание части предложения, по которому в сочетании с входным текстом, можно однозначно построить выходную семантическую модель, соответствующую анализируемому тексту. Выходная семантическая

модель представляется набором RDF триплетов, состоящих из субъекта, объекта и предиката.

Пример: пусть анализируемое предложение «Студент – это человек, который учится в университете». Входная схема, соответствующая этому предложению: «noun – это noun» , а выходная:

subject/##4##
object/##1##
property/#subClassOf

Полученный триплет:

object/Студент
property/#subClassOf
subject/человек

К достоинствам этого подхода можно отнести возможность его использования в системах семантического поиска, в том числе в ИПС общего назначения. К недостаткам же можно отнести то, что используемые лексико-синтаксические шаблоны учитывают только синтаксическое строение предложения и не опираются на семантику слов, поэтому при использовании данного подхода невозможно глубокое проникновение в семантику текста, а также велика вероятность ложных распознаваний отношений между понятиями.

2.6. Подход к извлечению фактов из текста на основе онтологии

Подход к извлечению фактов, предложенный И.С. Кононенко и Е.А. Сидоровой [10], предполагает использование онтологии предметной области, словарей предметной лексики, модели сегментации документов и схем извлечения фактов, которые связывают термины словаря с элементами онтологии. Особенность предлагаемого подхода состоит в том, что процесс анализа ведется под управлением онтологии, которая расширяется за счет полученной в результате анализа информации, что, в свою очередь, является основой пополнения лингвистической базы знаний. Технология ориентирована на анализ документов жанра деловой прозы, для которой характерны ограниченность предметной области и языка документов, наличие строгой модельной ситуации, четкость функций каждого сообщения.

Отличительной чертой предложенного подхода является ориентация на конкретные предметные знания. Выбор правил сборки словосочетаний и фактов определяется спецификой предметной области и структурой целевой онтологии. Онтология определяет то, какую именно информацию необходимо извлекать из текста документа. Результат анализа документа

представляется в виде семантической сети информационных объектов, являющихся экземплярами понятий и отношений, заданных онтологией предметной области. Для подхода характерно преимущественное использование лексико-семантической информации, что не исключает применения частичного синтаксического анализа и синтаксических ограничений, составляющие схем фактов.

Каждый факт с точки зрения рассматриваемого подхода имеет тип (название отношения) и список аргументов. Модель извлечения факта из текста должна учитывать множество языковых способов представления данного отношения в языке.

На характеристики аргументов факта накладываются следующие ограничения:

1. Морфологические и семантические (например, arg1. Падеж = рд, arg1. SemClass=Лок);
2. Синтаксической сочетаемости вершин синтаксических групп, представляющих аргументы схемы (например, Synt = Согл(число, падеж));
3. На взаиморасположение аргументов в тексте.

Пример типичной схемы:

Scheme Персона_с_инициалами
arg1: Term::ФИО(фио-тип: фам)
arg2: Term_lex::инициалы()
Condition Position = preposition_priority, Contact = absolute
ð Object::Персона(Фамилия: arg1.Name, Инициалы: arg2.Value)

Извлечение информации из текста

Процесс обработки текста включает этапы: графематический анализ, лексический анализ, сегментация, морфологический анализ, сборка фактов и формирование содержания документа.

Процесс извлечения фактов из текста базируется на схемах извлечения фактов. Например, извлечение из текста объекта класса *Персона*, представленного именной группой типа *ФИО*, демонстрируется схемой, приведенной выше. Аналогичным образом выглядят схемы для извлечения атрибутов и отношений.

После извлечения фактов из текста осуществляется генерация информационных объектов, соответствующих найденным фактам, и уточнения объектов и их характеристик путем взаимодействия с БД системы, а именно:

1. Слияние референтных объектов по принципу: если двум объектам в БД сопоставился один объект, то делается вывод о тождестве

данных объектов.

2. Уточнение неявно выраженных характеристик, например, если в отношении *Сотрудник* атрибут *Должность_роль* = «первое лицо», то определяется значение атрибута *Должность* на основании информации о типе *Организации* и названии руководящей должности для данного типа.

Достоинством и одновременно интересным свойством этого подхода является возможность автоматического расширения онтологии, на которую опирается система, в процессе анализа. Кроме того, при условии построения корректных онтологии и базы правил извлечения фактов данный подход гарантирует высокую точность извлечения фактов. Однако вследствие необходимости ручной коррекции базы правил и онтологии при их создании процесс разработки и поддержки такой системы извлечения трудоемок.

ЗАКЛЮЧЕНИЕ

В данной статье был приведен обзор некоторых методов формального представления знаний и семантики текстов, а также способов извлечения знаний из неструктурированных текстов, показавшихся автору наиболее интересными. Однако следует отметить, что, несмотря на разноплановость рассмотренных в статье методов, список таковых далеко не исчерпан.

СПИСОК ЛИТЕРАТУРЫ

1. Батура Т.В., Мурзин Ф.А. Машинно-ориентированные логические методы отображения семантики текста на естественном языке: моногр. / Институт систем информатики им. А.П. Ершова СО РАН. — Новосибирск: Изд. НГТУ, 2008. — 248 с.
2. Ермаков А.Е., Плешко В.В. Семантическая интерпретация в системах компьютерного анализа текста // Информационные технологии. – 2009. – N 6. – С. 27.
3. Симаков К. В. Модели и методы извлечения знаний из текстов на естественном языке // Автореферат диссертации на соискание ученой степени кандидата технических наук.: М. 2008.
4. Андреев А.М., Березкин Д.В., Симаков К.В. Метод обучения модели извлечения знаний из естественно-языковых текстов // Вестник МГТУ. Приборостроение. — 2007. — №3. — С. 75-94.
5. Sven Hartrumpf WOCADI [Electronic resource]. — Mode of access: http://pi7.fernuni-hagen.de/research/wocadi/wocadi_demo.html

6. Hermann Helbig Knowledge Representation with Multilayered Extended Semantic Networks (the MultiNet paradigm) [Electronic resource]. — Mode of access: http://pi7.fernuni-hagen.de/research/multinet/multinet_en.html
7. Hermann Helbig, Sven Hartrumpf Word Class Functions for Syntactic-Semantic Analysis [Electronic resource]. — Mode of access: http://pi7.fernuni-hagen.de/papers/helbig_hartrumpf97.pdf
8. Осипов Г.С., Смирнов И.В., Тихомиров И.А. Реляционно-ситуационный метод поиска и анализа текстов и его приложения // Искусственный интеллект и принятие решений [Электрон. ресурс]. — 2008. — N 2. — С. 3–10. — Режим доступа: http://www.raai.org/library/aidt/aidt2008-2/aidt2008-2.files/2008-02_3_10.pdf
9. Рабчевский Е., Крупов С., Рожков М., Булатова Г. Семантический анализ текста на основе лексико-синтаксических шаблонов для информационного поиска [Электрон. ресурс]. — 2009. — Режим доступа: http://rabchevsky.name/node/75#ir
10. Сидорова Е. А., Кононенко И. С. Подход к извлечению фактов из текста на основе онтологии. // Материалы международной конференции «Диалог 2009». — М. — С.451-458.

СОДЕРЖАНИЕ

CONTENTS

МОЛОДАЯ ИНФОРМАТИКА

СБОРНИК ТРУДОВ
АСПИРАНТОВ И МОЛОДЫХ УЧЕНЫХ

Под редакцией
к.ф.-м.н. А.Ю. Пальянова

Рукопись поступила в редакцию 20.10.11
Редактор Т.М. Бульонкова

Подписано в печать 21.12.11
Формат бумаги 60 × 84 1/16 Объем 6.4 уч.-изд.л., 7.0 п.л.
Тираж 75 экз.

Центр оперативной печати «Оригинал 2»
г. Бердск, ул. Островского, 55, оф. 02, тел. (383) 214-45-35