ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Вестник СибГУТИ № 3 (47) 2019

Выпускается ежеквартально, выходит с 2007 г.

Адрес редакции

630102, г. Новосибирск, ул. Кирова, д. 86 e-mail: vestnik@sibsutis.ru

Информация о журнале доступна в сети Internet по адресу http://vestnik.sibsutis.ru.

Журнал включён в Перечень ВАК российских рецензируемых научных журналов, в которых должны быть опубликованы основные научные результаты диссертаций на соискание учёных степеней доктора и кандидата наук.

Журнал зарегистрирован Федеральной службой по надзору за соблюдением законодательства в сфере массовых коммуникаций и охране культурного наследия, свидетельство о регистрации ПИ № ФС77-25835 от 29.09.2006. ISSN 1998-6920. Подписной индекс в объединённом каталоге «Пресса России» – 82519.

Отпечатано в издательском центре СибГУТИ. Бумага офсетная, формат А4. Тираж 300 экз.

СОДЕРЖАНИЕ

От главного редактора тематического выпуска

В данном выпуске журнала в виде научных статей опубликованы результаты исследований, предварительно представленные в избранных докладах семинара «Наукоёмкое программное обеспечение», который проходил в рамках Ершовской конференции по информатике в июле 2019 года в Новосибирске.

Ершовская конференция по информатике – это один из главных форумов России, посвященных исследованиям и приложениям в таких областях информатики, как компьютерные науки, методология и технология программирования, информационные технологии. Конференция регулярно проводится с 1991 года и посвящается выдающемуся отечественному ученому, специалисту в области теоретического и системного программирования академику Андрею Петровичу Ершову (1931–1988). На конференциях собираются ученые, разработчики и пользователи программного обеспечения для того, чтобы представить и обсудить самые последние новшества, идеи, тенденции и результаты исследований в направлениях, обозначенных в тематике конференции.

В рамках Ершовской конференции по информатике обычно проводится несколько рабочих семинаров. В 2019 году ученые, представители высокотехнологичных компаний и пользователи программного обеспечения могли принять участие в трех семинарах:

1. Наукоёмкое программное обеспечение.
2. Информатика образования.
3. Семантика, спецификация и верификация. Теория и приложения.

Семинар «Наукоёмкое программное обеспечение» посвящён широкому кругу вопросов создания прикладного программного обеспечения. В ходе его работы были представлены новые программные продукты и системы, которые стали предметом дискуссии представителей фундаментальной и прикладной науки с разработчиками и заказчиками научного и наукоёмкого программного обеспечения. Наиболее значительные доклады были рекомендованы программным комитетом семинара для подготовки на их основе статей для данного тематического выпуска.

Редактор тематического выпуска д.ф.-м.н., профессор А. Г. Марчук

УДК 004.93.11

Методы обнаружения и выделения областей на текстурных изображениях

Г. Б. Абдикеримова, А. Л. Бычков, Вей Синьюй, Ф. А. Мурзин, Н. Е. Русских, Е. И. Рябчикова, С. С. Хайрулин[1]

В статье речь идет о методах анализа текстурных изображений. Рассматриваются микрофотографии растительного сырья, полученные просвечивающей электронной микроскопией. Работа выполнена для Института химии твердого тела и механохимии СО РАН. Основной целью исследований является разработка и реализация алгоритмов, позволяющих обнаруживать и выделять на изображении области, представляющих интерес для специалистов-химиков. Например, области, в которых происходит разупорядочение исходной структуры материала после применения различных механохимических методов обработки. Для решения поставленной задачи используются текстурные признаки, кластеризация, R/S-анализ, ортогональные преобразования, вейвлет-анализ. Большое внимание было уделено разработке программных инструментов, позволяющих осуществлять выбор признаков, описывающих текстурные различия, чтобы сегментировать текстурные области на подобласти. То есть исследуется вопрос о применимости наборов текстурных признаков и других параметров для анализа экспериментальных данных с целью выявить на микрофотографиях характерные участки, которые в будущем можно будет увязать с пористостью, химической реактивностью и т.д.

Ключевые слова: обработка изображений, микрофотографии, текстурные признаки, кластеризация, R/S-анализ, ортогональные преобразования, электронная микроскопия, растительное сырье.

1. Введение

Исследования направлены на создание научно-технического задела в области обработки изображений текстурного типа. Целью проводимых исследований является разработка и поиск алгоритмов анализа изображений, полученных из различных источников, например, при помощи современных электронно-микроскопических методов. В том числе анализируются изображения ультраструктуры растительных клеточных стенок, полученные просвечивающей электронной микроскопией растительного сырья, после различной физико-химической и/или механохимической обработки. В настоящее время данная информация востребована учеными из различных областей – химиками, биологами, технологами, но обрабатывается на качественном (редко – полуколичественном) уровне вручную. Переход к алгоритмам, позволяющим оперировать большими объёмами данных, позволил бы перечисленным областям науки сделать значительный шаг вперёд, к совершенствованию существующих и созданию новых технологических процессов.

В работе используются различные наборы текстурных признаков (около 20) и спектральные преобразования на основе ортогональных матриц (наибольший интерес представляют 6

[1] Работа поддержана Российским научным фондом (грант № 16-13-10200) и Российским фондом фундаментальных исследований (грант № 18-08-01284).

преобразований). Также проводились эксперименты по применению R/S-анализа и вейвлет-анализа.

Программные продукты, позволяющие детально анализировать текстуры, могут успешно применяться в различных областях науки и промышленности. Прежде всего, это химия и материаловедение. Можно анализировать материалы органического происхождения, срезы металлов и минералов, керамику и др.

Спектр задач не ограничивается анализом микрофотографий. Так, при обработке аэрокосмических снимков исследователи также имеют дело с различными текстурами. По текстурным признакам можно определить хвойный или лиственный лес, поля, засеянные зерновыми или бобовыми растениями, и др. Также можно выделить лес, пораженный вредителями, опустыненные территории. Другая область исследований, где эти методы могут быть эффективно использованы, – это диагностика внутренних патологий человека, в том числе злокачественных, методом анализа изображений, полученных с помощью тепловизора.

Принципиальное отличие идей проекта от существующих аналогов состоит в корректном применении математических методов и в более глубокой их проработке. Например, текстурных признаков известно более двухсот, в научных обзорах приводят обычно около пятидесяти. В то же время на практике, как правило, используют 3–4 признака, например, при обработке космических снимков. То есть исходные изображения остаются не исследованными в полной мере. То же самое можно сказать о применении интегральных преобразований. Например, при исследовании прочности металлов под нагрузками используют преобразование Хаара, чтобы охарактеризовать трещиноватость. Вопрос, какую информацию можно получить на основе других преобразований, почти не изучен. В литературе по химии древесины говорится о полезности R/S-анализа и фрактального анализа для соответствующих исследований, но сведения носят отрывочный характер.

2. Методы анализа текстурных признаков

Несмотря на повсеместное присутствие текстур в изображениях, единого и формального подхода к описанию текстуры и строгого её определения на данный момент не существует. Методы анализа текстур, как правило, разрабатываются для каждого отдельного случая.

В [1] под текстурой понимают «пространственную организацию элементов в пределах некоторого участка поверхности». Там же объясняется, что эта организация обусловлена определенным статистическим распределением интенсивности серых тонов или тонов различного цвета. Участок может считаться текстурным, если количество отмечаемых на нем перепадов интенсивности или изменений цвета достаточно велико. В [2] текстурой называют «некоторым образом организованный участок поверхности». В [3] текстура определяется как матрица или фрагмент пространственных свойств участков изображений с однородными статистическими характеристиками.

Текстуры можно разделить на несколько классов следующим образом:

1) по происхождению: на искусственные – например, графические узоры, и естественные – например, трава, лес, земля;

2) по структуре поверхности: структурные, состоящие из геометрически правильных повторяющихся элементов, и стохастические, сформированные последовательностью случайных элементов; по относительным размерам элементов текстуры: мелкозернистые и крупнозернистые;

3) по форме элементов текстуры: волнистые, пятнистые, неправильные, линейчатые и так далее [1].

Из приведенных выше определений и характеристик следует, что текстура – это некоторый участок изображения, тот, который имеет однородные статистические характеристики. Это значит, что каждую текстуру данного класса можно описать с помощью характерного свойства, общего для всех текстур данного класса [1]. Такие свойства называют текстурными

признаками [4]. Текстурные признаки играют важную роль при разделении изображения на отдельные области.

В качестве таких признаков можно использовать статистические характеристики пространственных распределений, вычисляемые как меры однородности по одномерной гистограмме значений сигналов (характеристики 1-го порядка) и по двумерным гистограммам значений сигналов (характеристики 2-го порядка).

Второй класс – это признаки, учитывающие взаимное расположение. Для формирования текстурных признаков, учитывающих взаимное расположение пикселей внутри скользящего окна, применяется подход, основанный на использовании матрицы смежности (другое название – матрица распределения градиентов [4]).

В рассматриваемой задаче исследуются серые (полутоновые) изображения. Таким образом, изображение задается в виде матрицы, значениями которой являются значения яркости пикселей в интервале от 0 до 255.

Стандартный подход для вычисления текстурных признаков следующий. Необходимо выбрать так называемое бегущее окно с нечетной стороной: 3, 5, 7 пикселей. Признак вычисляется внутри бегущего окна. Размер локального фрагмента является носителем текстурных свойств. Значение признака записывается в новую матрицу того же размера, что и исходная. В новой матрице значение записывается в точку с координатами, равными координатам центра бегущего окна. Элементы новой матрицы получаются в некотором интервале [A, B]. Далее обычно этот интервал линейно отображается в отрезок [0, 255]. После этого имеется возможность визуализировать результат вычисления текстурного признака.

Эксперименты показали, что стандартный подход в нашем случае малоинформативен. Поэтому было решено использовать нестандартный подход. А именно, текстурные признаки вычисляются по большим окнам (в том числе по неквадратным), которые пользователь может задавать, выбирая область, которая может представлять для него интерес. То есть речь идет о вычислении числовых характеристик, относящихся к обширным областям, включающим различного рода артефакты.

Рассмотрены 18 наиболее важных текстурных признаков, все они реализованы в программе. Ниже в таблице приведены некоторые из них.

Таблица 1. Примеры текстурных признаков

Признаки, основанные на статистических характеристиках	Признаки, учитывающие взаимное расположение
k-th начальный момент $T_1^k = n^{-2}\sum_{i=1}^{n}\sum_{j=1}^{n}[f(i,j)]^k$.	Среднее $T_1 = \mu_i = \mu_j = \sum_{i=0}^{N-1}\left[i\sum_{j=0}^{N-1} P(i,j)\right]$.
Энтропия $T_2 = -\sum_{g=0}^{N-1} F(g)\log_{10} F(g)$.	Энергия $T_2 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[P(i,j)]^2$.
Энергия $T_3 = \sum_{k=0}^{N-1}[F(g)]^2$.	Вариация $T_3 = \sigma_i^2 = \sum_{i=0}^{N-1}\left[(i-\mu_2)^2\sum_{j=0}^{N-1} P(i,j)\right]$.
Вариация $T_4 = -\sum_{g=0}^{N-1}(g-\mu)^2 F(g)$.	Однородность $T_4 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} P(i,j)/(1+\lvert i-j\rvert)$.

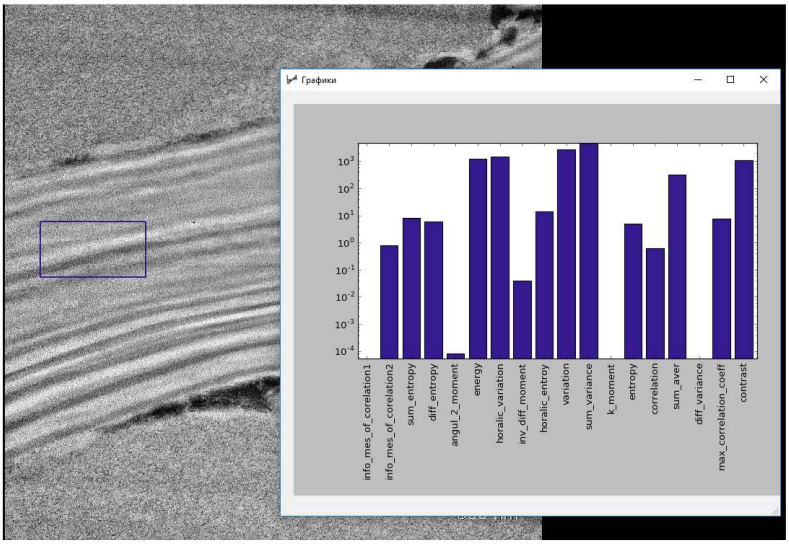


Рис. 1. Вычисление текстурных признаков

3. Алгоритмы кластеризации

Сегментация изображения может быть осуществлена локальным и глобальным способом. Локальный способ оценивает наличие границы между областями по поведению признаков в окрестности точки изображения. Глобальный способ предполагает предварительную кластеризацию пространства признаков и затем установление соответствия между пикселем изображения и кластером, в который попадает его вектор признаков [5].

Одни из эвристических методов кластеризации – методы, основывающиеся на последовательной агломеративной процедуре. Достоинством этих методов является простота вычислительной процедуры и алгоритмов.

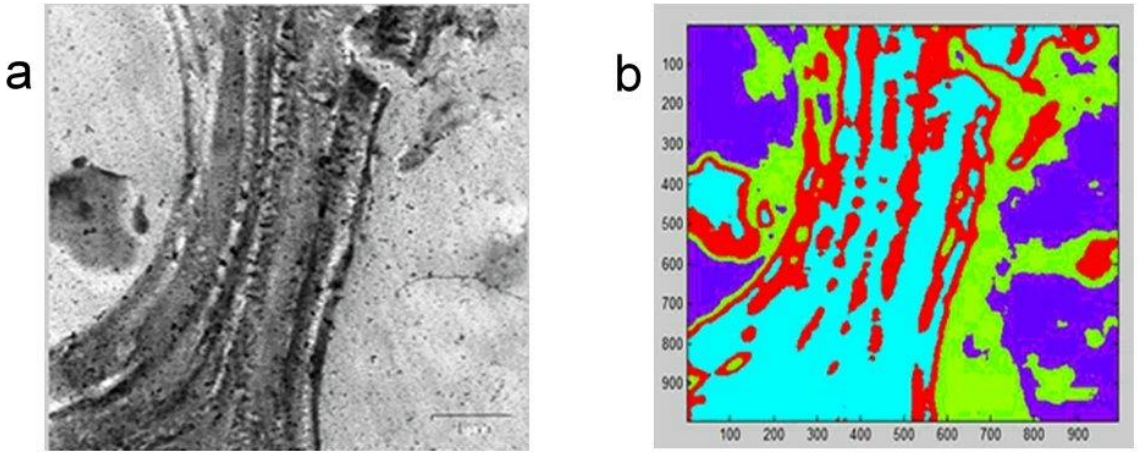


Рис. 2. (a) исходное изображение; (b) результат сегментации

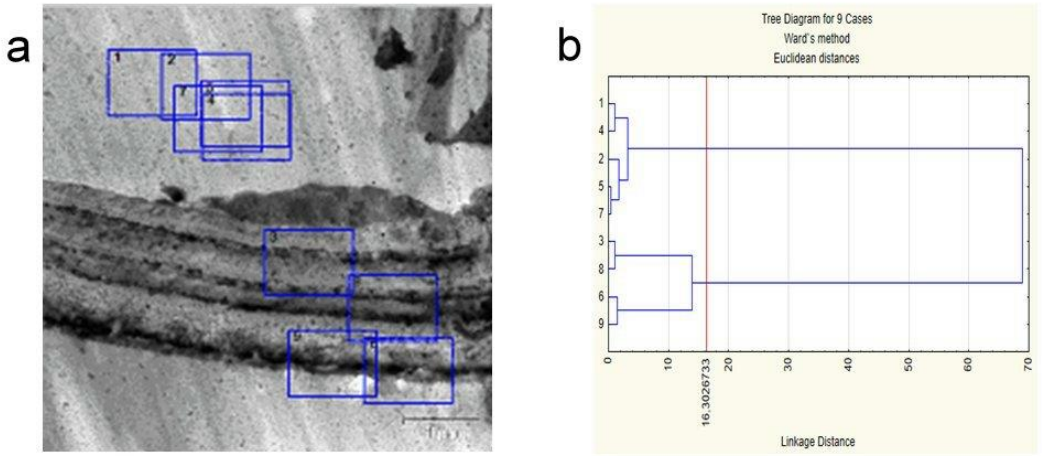


Рис. 3. (a) выбор фрагментов изображения; (b) иерархическая кластеризация

4. R/S-анализ для изображений

Метод R/S-анализа – это статистический метод анализа временных рядов или числовых последовательностей (преимущественно финансовых), позволяющих определить некоторые важные их характеристики, такие как наличие непериодических циклов, «памяти» у процесса, степень хаотичности и т.д. [6]. Метод применим и для анализа изображений, т.к. можно рассматривать последовательность значений функции яркости вдоль некоторой прямой или кривой линии. Важной характеристикой, вычисляемой методом R/S-анализа, является показатель Хёрста, обычно обозначаемый H, который характеризует степень хаотичности процесса [7]. Показатель Хёрста также называют иногда фрактальной размерностью. Интересно, что метод применяется в химии для анализа микрофотографий (например, лигнина) [8, 9] и он показал определенную эффективность.

Метод программно реализован в среде MATLAB, проводится анализ изображений. Изображения программными средствами делится на отдельные участки. Если участок изображения принадлежит так называемой области скейлинга, то это позволяет выявить с помощью оценки фрактальной размерности соседних участков более мелкие электроноплотные (чёрные) образования, которые, скорее всего, являются остатками внутриклеточного содержимого липидной или белковой природы (дробный метод исследования). При этом фрактальная размерность участков микрофотографий клеточных стенок (например, соломы пшеницы) будет лежать в интервале нормы, а фрактальная размерность участков с лигнифицированными слоями – ниже границы нормы. Было произведено разделение изображения на 1024 части. Для каждой части изображения выполнялся расчет фрактальной размерности.

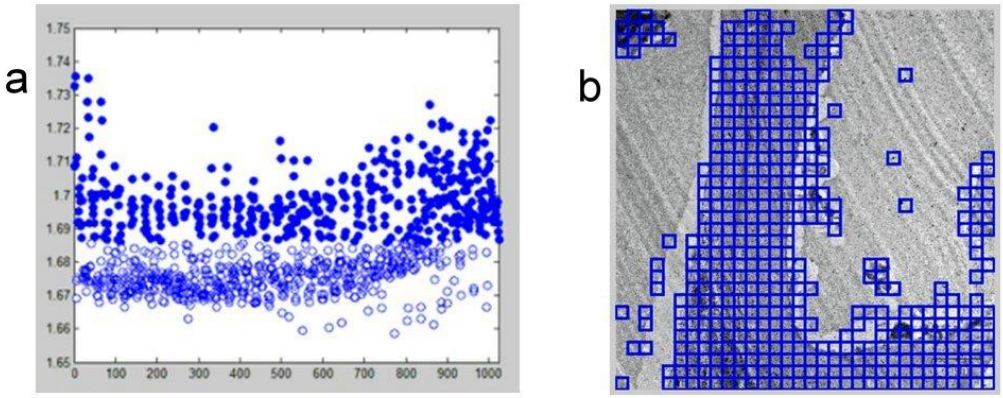


Рис. 4. (a) значения показателя Хёрста; (b) выделенная область

Результаты применения дробного метода изображаются в виде множества кружков, отражающих вычисленную фрактальную размерность. Синие кружки соответствуют синим квадратам на исходном изображении, белые кружки указывают на фон (рис. 4).

5. Ортогональные преобразования

Спектральный анализ является мощным инструментом анализа сигналов и изображений. Уже давно замечено, что спектр очень чутко реагирует на различные изменения в структуре сигналов и изображений [10].

Для проведения спектрального анализа необходимо предварительно разложить сигнал или изображение по частотам. Для этого применяются [11] различные наборы базисных функций. Соответствующие алгоритмы называют преобразованиями: косинусное, Адамара, Хаара, наклонное и др. Отметим, что преобразования Хаара и Добеши являются простейшими вейвлет-преобразованиями.

Программа реализована в среде MATLAB и позволяет осуществлять спектральные преобразования шести видов: 1) косинусное, 2) Адамара порядка 2^n, 3) Адамара порядка $n = p+1$, $p \equiv 3 (\mathrm{mod}\, 4)$ – простое число, т.е. на основе символа Лежандра, 4) Хаара, 5) наклонное, 6) Добеши-4. Пример преобразования приведен ниже (рис. 5).

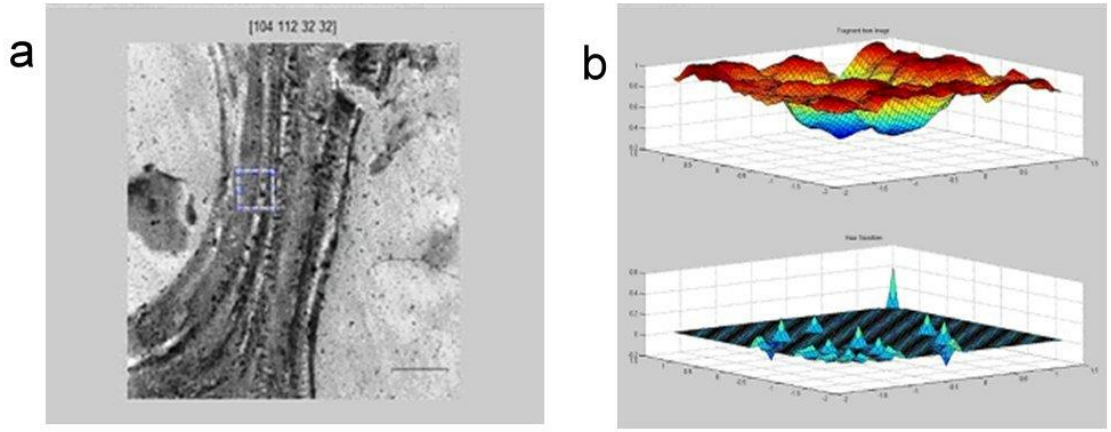


Рис. 5. (a) выделенное окно; (b) функция яркости в окне и результат преобразования Хаара

Для изучения текстур можно применять разные нестандартные подходы с использованием ортогональных преобразований. Например, исходное изображение разбивается на непересекающиеся квадратные окна. Как показывают эксперименты, целесообразно взять размер окна достаточно большим, например, 32 × 32, 64 × 64 и т.д. Затем в каждом окне мы делаем ортогональное преобразование. Далее некоторые спектральные коэффициенты могут быть отброшены, например, высокочастотные или в нескольких определенных частях спектра. В двумерном случае частотные спектры представляют собой двумерные матрицы. Естественно, что можно расположить элементы матриц в векторах. Например, строки матрицы могут быть расположены последовательно одна за другой. Фактически позиции в векторе, в которых расположены спектральные коэффициенты, которые мы обнулили, могут быть вычеркнуты из результирующего вектора. Далее процедура кластеризации может быть выполнена применительно к полученным векторам.

Другой подход – это вейвлет-декомпозиция [12]. Можно сказать, что вейвлет-разложение [8] выполняется с помощью ассоциированных с деревом блоков двухканальных фильтров. В вершине этого дерева, высота которого равна d, должен находиться сигнал длины, равной

2^d. Соответственно, размер изображения должен быть $2^d \times 2^d$. Если это нарушено, то недостающие значения обычно добавляются равные нулю.

На каждом шаге преобразования изображение разбивается на 4 матрицы. Одна из них представляет собой изображение, похожее на оригинал, но оно меньше по горизонтали и по вертикали в 2 раза. Можно сказать, что это «грубая» версия исходного изображения. В трех других матрицах «кодируются» различные перепады яркости исходного изображения. Таким образом производится «детализация» информации, содержащейся в изображении.

Значения элементов могут не попадать в интервал {0,…, 255}. Для каждой из матриц мы находим значения минимального и максимального элементов и отображаем соответствующий интервал в интервал {0,…, 255}. Это позволяет визуализировать результаты в виде серых или цветных изображений.

Простейшими формами вейвлет-преобразований для изображений являются преобразования Хаара и Добеши-4. Ниже приведены результаты такого рода эксперимента для преобразования Хаара (рис. 6).

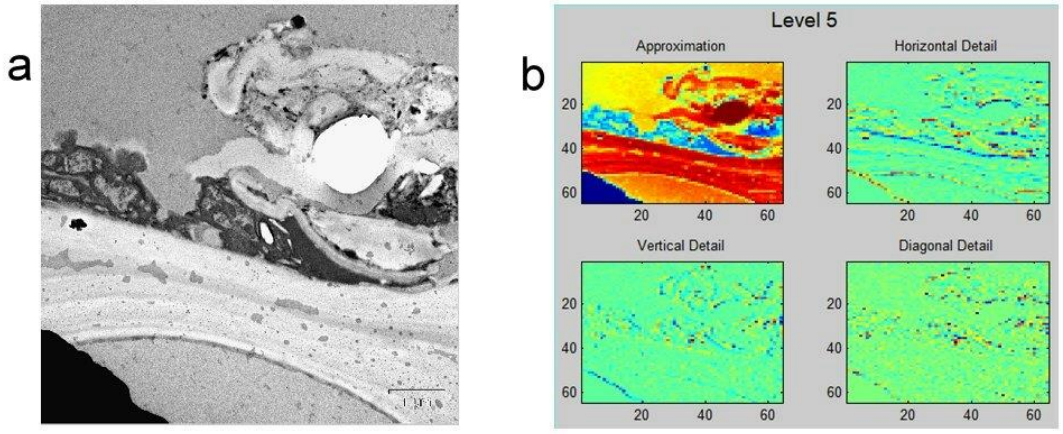


Рис. 6. (a) исходное изображение;
(b) 5-й уровень вейвлет-декомпозиции посредством преобразования Хаара

6. Заключение

Работа посвящена изучению текстурных изображений, сегментации текстур, ортогональным преобразованиям, R/S-анализу и т.д. Исследуются микрофотографии растительного сырья, измельченного на специальных мельницах и подвергнувшегося нагреванию или охлаждению, в том числе до криогенных температур. Работа выполнена для Института химии твердого тела и механохимии СО РАН.

В основном речь идет о расчете различных видов текстурных признаков и других характеристик изображений. Был проанализирован большой набор методов исследования текстур, из рассмотренных методов были выбраны наиболее информативные. В итоге были реализованы следующие программные инструменты:

1. Программа для расчета различных текстурных признаков.

2. Программы для кластерного анализа в стандартном и иерархическом вариантах. Были проведены эксперименты. В первом случае использовалось 18 текстурных признаков с окном прокрутки 9×9 пикселей. Во втором – 5 текстурных признаков с окном 400×300 пикселей. Точнее, во втором случае после проведения многочисленных экспериментов были выбраны пять наиболее информативных текстурных признаков, по которым и была проведена кластеризация.

3. Прототипы программ, использующих R/S-анализ. Эксперименты показали интересные результаты, являющиеся полезными на практике. Фон и клеточная стенка четко отделяются

друг от друга. Отметим, что подобные высказывания встречаются в научной литературе по изучению лигнина, но без особых деталей.

4. Программа, позволяющая осуществлять спектральные преобразования шести видов. Рассмотрены некоторые их приложения. Например, вейвлет-декомпозиция изображений.

В дальнейших исследованиях векторы показателей могут быть связаны с химической реактивностью, пористостью и другими характеристиками. Программная система может быть обучена на примерах с использованием алгоритмов, основанных на нейрокомпьютерных или других подходах, обычно применяемых в машинном обучении. После обучения система сможет предсказывать значения параметров.

Литература

1. *Андреев Г. А., Базарский О. В., Глауберман А. С., Колесников А. И., Коржик Ю. В., Хлявич Я. Л.* Анализ и синтез случайных пространственных текстур // Зарубежная радиоэлектроника. 1984. № 2. С. 3–33.
2. *Харалик Р. М.* Статистический и структурный подходы к описанию текстур // ТИИЭР. 1979. Т. 67, № 5. С. 98–119.
3. *Потапов А. А.* Новые информационные технологии на основе вероятностных текстурных и фрактальных признаков в радиолокационном обнаружении малоконтрастных целей // Радиотехника и электроника. 2003. Т. 48, № 9. С. 1101–1119.
4. *Колодникова Н. В.* Обзор текстурных признаков для задач распознавания образов // Доклады ТУСУРа. Автоматизированные системы обработки информации, управления и и проектирования. 2004. С. 117–118.
5. *Sidorova V. S.* Hierarchical Cluster Algorithm for Remote Sensing Data of Earth // Pattern Recognition and Image Analysis. 2012. V. 22, № 2. P. 373–379.
6. Федер Е. Фракталы. М.: Мир, 1991. 192 с.
7. *Петерс Э.* Хаос и порядок на рынках капитала. М.: Мир, 2000. 90 с.
8. *Карманов А. П.* Лигнин. Структурная организация и самоорганизация // Тезисы докладов III Всероссийской конференции «Химия древесины и органический синтез». Институт химии Коми научного центра Уральского отделения РАН, Сыктывкар, 1999. С. 66–67.
9. *Карманов А. П., Матвеев Д. В.* Проблемы химии древесины и лесохимии // Институт химии Коми научного центра Уральского отделения РАН, Сыктывкар. 2001. С. 50–52.
10. *Rani R.* Performance analysis of different orthogonal transform for image processing application // Inter. J. of Applied Research. 2015. V. 1, № 12. P. 844–847.
11. *Shahdoosti H. R., Mirzapour F.* Spectral-spatial feature extraction using orthogonal linear discriminant analysis for classification of hyperspectral data // European J. of Remote Sensing. 2017. V. 50, № 1. P. 111–124.
12. *Воробьев В. И., Грибунин В. Г.* Теория и практика вейвлет-преобразования. Санкт-Петербург: Изд.-во Военного университета связи, 1999. 92 с.

Статья поступила в редакцию 27.08.2019.

Абдикеримова Гульзира Бахытбековна
ст. преподаватель, Евразийский национальный университет им. Л. Н. Гумилева (010000, Казахстан, Нурсултан, ул. Сатбаева, 2), e-mail: gulzira1981@mail.ru.

Бычков Алексей Леонидович
к.х.н., с.н.с., Институт химии твердого тела и механохимии СО РАН (630128, Новосибирск, ул. Кутателадзе, 18), e-mail: bychkov.a.l@gmail.com.

Вей Синьюй
аспирант, Хэйлунцзянский университет (150000, Харбин, Китай, ул. Сюефу 74), e-mail: xinyuwei2016@163.com.

Мурзин Федор Александрович
к.ф.-м.н., зам. директора по научной работе, Институт систем информатики им. А. П. Ершова СО РАН (630090, Новосибирск, пр. Академика Лаврентьева, 6), e-mail: murzin@iis.nsk.su.

Русских Николай Евгеньевич
аспирант, Институт систем информатики им. А. П. Ершова СО РАН, e-mail: russkikh.nikolay@gmail.com.

Рябчикова Елена Ивановна
д.б.н., профессор, руководитель группы микроскопических исследований, Институт молекулярной биологии и фундаментальной медицины СО РАН (630090, Новосибирск, пр. Академика Лаврентьева, 8), e-mail: lenryab@niboch.nsc.ru.

Хайрулин Сергей Сергеевич
м.н.с., Институт систем информатики им. А. П. Ершова СО РАН, e-mail: s.khayrulin@gmail.com.

Methods for detecting and highlighting areas in textural images

G. B. Abdikerimova, A. L. Bychkov, Wei Xinyu, F. A. Murzin, N. E. Russkikh, E. I. Ryabchikova, S. S. Khayrulin

The article deals with methods for analyzing textural images. Micrographs of plant materials obtained by the transmission electron microscopy are considered. This work was carried out for the Institute of Solid State Chemistry and Mechanochemistry SB RAS. The main goal of the research is the development and implementation of algorithms allowing us to detect and highlight areas of interest to chemists in the image. For example, areas, in which the initial structure of the material is disordered after applying various mechanochemical methods of processing. To solve this problem, we use: analysis of textural features, clustering, R/S-analysis, orthogonal transformations, wavelet analysis. Much attention was paid to the development of software tools that allow us to select features describing textural differences in order to segment textural regions into subregions. So, the question of the applicability of sets of textural features and other parameters for the analysis of experimental data is being investigated in order to identify characteristic areas in microphotographs that can be associated in future with the porosity, chemical reactivity, etc.

Keywords: image processing, microphotographs, textural features, clustering, R/S-analysis, orthogonal transformations, electron microscopy, plant materials.

УДК 519.685+519.682.2+004.032.22

Подход к построению системы детерминированного параллельного программирования на основе монотонных объектов

А. И. Адамович, А. В. Климов

В связи с взрывным ростом сложности программ для многоядерных процессоров и суперкомпьютеров в последние десятилетия приобретает популярность и становится всё более актуальной идея параллельных вычислений с детерминированностью, гарантированной языком и системой программирования. В статье анализируется проблема, как сделать параллельное программирование как можно более детерминированным, а также некоторые существующие подходы к её решению. Описываются принципы построения системы объектно-ориентированного программирования, разрабатываемой авторами, предоставляющей возможность писать как детерминированный, так и недетерминированный код с гарантиями прикладному программисту, что его программа будет детерминированной. Система и входной язык имеют два уровня: верхний – для пользователей, разрабатывающих прикладные программы; нижний – для разработчиков библиотек классов, называемых *монотонными*. Входной язык подсистемы верхнего уровня похож на функциональный язык с возможностью создания и использования неизменяемых и монотонных объектов. Библиотеки монотонных классов гарантируют, что все программы на подъязыке верхнего уровня, использующие только монотонные классы, являются детерминированными и идемпотентными при их распараллеливании асинхронными вызовами всех функций. Обсуждаются показательные задачи, реализуемые на данной системе.

Ключевые слова: модели параллельных вычислений, детерминированные программы, функциональное программирование, объектно-ориентированное программирование, монотонные объекты.

1. Введение

Параллельные и конкурентные (*англ*. concurrent) программы в общем случае являются недетерминированными – дают разные результаты при нескольких прогонах, так как для эффективной реализации на современной аппаратуре требуется явное использование таких средств программирования, как процессы, потоки, треды (*англ*. threads), читающие и изменяющие общие ресурсы и дающие разные результаты при различном порядке доступа потоков к ресурсам. Отладка, модификация и сопровождение таких программ намного более трудоемки, чем привычные, для массовых программистов детерминированных последовательных программ. Поэтому многие языки высокого уровня «прячут» от «обычного» программиста средства конкурентного программирования на уровень реализации и в библиотеки, разрабатываемые экспертами. Однако такие решения ограничивают изобразимые на этих языках классы программ и вынуждают писать менее эффективные приложения, не масштабируемые при увеличении числа процессоров, ядер и других аппаратных средств.

Таким образом, мы имеем следующую ситуацию. Во-первых, детерминированные языки параллельного и конкурентного программирования существуют и развиваются (в качестве

яркого примера приведем чисто функциональный язык Haskell и его библиотеки для параллельного программирования [20]). Во-вторых, нет и не может быть одного языка или библиотеки детерминированного параллельного программирования, который удовлетворил бы все потребности, даже если зафиксировать круг языковых понятий конкурентного программирования, например, объектно-ориентированные языки типа Java с понятием тредов. В результате значительно разнесены уровень детерминированного программирования, считающийся «высоким», и уровень реализации языков и библиотек, считающийся «низким». На этих уровнях используются сильно различающиеся языки и инструменты и требуется очень разная квалификация разработчиков – столь же отличающиеся, как, например, у разработчиков систем программирования и их пользователей.

Возникает вопрос: насколько можно приблизить эти уровни? Нельзя ли в рамках одного языка программирования (пусть это будет, скажем, объектно-ориентированный язык типа Java или Kotlin [9]) дать и универсальные средства недетерминированного программирования для создания базовых инструментов и библиотек, и определить «высокий» уровень в виде подъязыка, проверяемого компилятором, при программировании на котором гарантируется детерминированность. «Нижний», универсальный, уровень потребуется для постоянного расширения и развития предметно-ориентированных библиотек для реализации определенных типов параллельных алгоритмов. Для разработки таких библиотек требуются повышенные трудозатраты, но потом библиотеки используются в большом числе прикладных программах данного класса, программирование, отладка и сопровождение которых становятся намного легче и дешевле благодаря детерминированности, гарантированной библиотеками и системой программирования.

Авторы данной статьи ставят целью дать положительный ответ на этот вопрос в виде двухуровневой системы программирования на языке типа Java. Нижний, базовый, уровень – это весь язык Java, Kotlin [9] или другой, на котором определяются классы, используемые на верхнем уровне. Верхний уровень – это Java-подобный язык или подмножество языка Java, близкое к чисто функциональному языку с естественной (для таких языков) параллельной реализацией и с объектно-ориентированными расширениями, обеспечивающими использование классов нижнего уровня, не нарушая детерминированности параллельных вычислений. Кроме того, мы требуем выполнения еще одного свойства программ – *идемпотентности,* предоставляющего возможность повторного вычисления любого выражения. Библиотечные классы нижнего уровня и их объекты, удовлетворяющие этим требованиям, мы называем *монотонными*. Данный проект продолжает наши работы по Т-системе и монотонным объектам [1–6, 8, 10, 15]. Эта статья является расширенной версией доклада [6].

Основные результаты данной статьи следующие:

- утверждается и демонстрируется на примерах возможность построения открытой системы параллельного программирования на базе объектно-ориентированных языков, в которой прикладному программисту представляется подмножество входного языка, гарантирующее детерминированность и идемпотентность всех его программ при использовании библиотек классов, созданных экспертами и называемых *монотонными*;
- дано формальное операционное определение понятия монотонных классов и объектов (продолжая наши предыдущие работы [5, 6, 8, 15]);
- приведен пример на языках Java и Kotlin определения монотонного объекта, реализующего понятие I-структуры [11], и пример вызывающей его программы на языке Kotlin с использованием средств параллельного программирования на сопрограммах;
- кратко охарактеризованы вопросы программирования на монотонных объектах двух классов задач: порождение эффективного объектно-ориентированного представления графов (невозможного на чисто функциональных языках) и оптимизационных переборных алгоритмов типа метода ветвей и границ.

Нам не известны работы по построению аналогичной системы детерминированного и идемпотентного объектно-ориентированного программирования.

Статья имеет следующее содержание. В разделе 2 дается обзор некоторых существующих подходов к детерминированности программ, которые ближе всего перекликаются с нашим подходом. В разделе 3 описывается двухуровневая архитектура системы детерминированного параллельного программирования. Раздел 4 объясняет, зачем к требованию детерминированности мы добавляем требование идемпотентности. В разделе 5 дается определение монотонных классов и объектов – ключевого понятия нашей работы. В разделе 6 приведен пример монотонного объекта, реализующего понятие I-структуры [11], и функции Фибоначчи, запрограммированной с его использованием. Раздел 7 содержит общую характеристику двух классов задач, на которых в настоящее время отрабатывается система и разрабатывается библиотека монотонных классов. Раздел 8 содержит заключение.

2. Близкие работы по детерминированному программированию

Тема детерминированности параллельных программ в последние десятилетия становится всё более популярной и актуальной в связи с всё большим распространением параллельных вычислительных устройств: как многоядерных центральных процессоров (CPU, central processing unit), так и разнообразных ускорителей – от графических (GPGPU, general purpose graphics processor unit) до программируемых логических интегральных схем (ПЛИС, FPGA, field-programmable gate array). Параллельное программирование становится всё более массовым, и возникает проблема его удешевления и повышения надежности. В статье [5] дан обзор некоторых методов и средств детерминированного программирования. В этом разделе приведены те из них, которые имеют б*о*льшее отношение к решаемой нами задаче.

2.1. Детерминированный параллелизм чисто функциональных языков

Отправной точкой, общей основой многих языков со средствами параллельного исполнения являются функциональные языки, которые в «чистом» виде не имеют побочных эффектов и потому считаются «легко» распараллеливаемыми. Распараллеливание кода на функциональном языке, конечно, подразумевает сохранение семантики языка, то есть эквивалентность результата параллельного исполнения эталонному последовательному, а также денотационной семантике, и тем самым – детерминированность.

Максимально параллельное исполнение функциональной программы делается вызовами каждой функции в отдельном параллельном процессе. Это отнюдь не является эффективным решением в общем случае: возникает задача, наоборот, ограничивать параллелизм, чтобы эффективно использовать аппаратные ресурсы, то есть не «распараллеливать», а «секвенциализировать» (*англ*. sequentialize), объединять потенциально параллельные группы вычислений в последовательный поток управления, тред. Прагматичное решение (которому мы следуем в нашем проекте) – дать программисту языковые средства обозначать, где параллельные вызовы, а где последовательный код. Если двигаться с другой стороны – от последовательных языков к параллельным, то аналогичное решение наблюдаем, когда вводится конструкция (или библиотечные средства), называемая в ряде языков «future» и «promise»: вызов функции в отдельном процессе, исполняемом параллельно до тех пор, пока вызвавшему процессу не понадобится результат и он не встанет на ожидание завершения вызванной функции.

Большинство языков, называемых функциональными, – «грязные», то есть разрешают побочные эффекты и не прячут понятие параллельного процесса, треда, позволяя явно управлять ими так же, как и в объектно-ориентированных языках со средствами параллелиз-

ма. Из распространенных функциональных языков лишь Haskell старается сохранять «чистоту» и осторожно расширяется средствами параллельных вычислений [20].

Среди большого разнообразия публикаций по этому направлению особенно интересен сборник, подводящий итоги по состоянию на конец 1990-х годов [14]. Не выходя за рамки функциональной модели вычислений, также строятся проблемно-ориентированные декларативные языки для конкретных областей применения. Например, таким является язык Норма [7] для решения сеточных задач некоторого класса из математической физики.

С другой стороны, современные объектно-ориентированные языки содержат в качестве своего подмножества чисто функциональный язык, на котором можно программировать в функциональном стиле без побочного эффекта. Таковы Java, C#, Kotlin, JavaScript и другие. Здесь основное неудобство – то, что компилятор не проверяет принадлежность к функциональному подмножеству и соблюдение этого условия – дело программиста.

2.2. Неизменяемые объекты, immutability

В мире объектно-ориентированных языков есть ряд работ по введению ограничений, проверяемых компилятором и/или во время исполнения и обеспечивающих нужные свойства. Если потребовать, чтобы все объекты были неизменяемыми (*англ*. immutable), то есть их состояние не менялось после отработки инициализаторов, то объектно-ориентированный язык практически превращается в функциональный, которому присущ детерминированный параллелизм.

На практике трудно обходиться совсем без изменений состояния и побочных эффектов, поэтому понятию неизменяемости (*англ*. immutability) придают различную степень «строгости». Статья [16] содержит обзор работ по этой теме.

2.3. Статические методы обеспечения детерминированности

Имеется много работ по статическому анализу кода с побочными эффектами, имеющему целью выявлять случаи, когда параллельное исполнение сохраняет детерминированность результата из-за отсутствия «гонок». Здесь особо отметим разработку языка Deterministic Parallel Java, DPJ [12] как имеющую прагматическую цель.

В наших работах эти результаты не используются, мы подходим к задаче с другой стороны: отказываясь от сложного статического анализа полного объектно-ориентированного языка мы накладываем на этот язык синтаксические ограничения, выделяя функциональное подмножество, и реализуем в монотонных классах динамические проверки необходимых по их семантике условий.

2.4. Обеспечение детерминированности операциями над данными

Следующий подход к обеспечению детерминированности параллельных программ, в отличие от методов предыдущего раздела, не использует никаких статических средств анализа. За основу берется чисто функциональный язык программирования, быть может, со всевозможными расширениями, не нарушающими функциональность и распараллеливаемость. Затем для взаимодействия параллельных процессов предоставляются специальные структуры данных с операциями, определенными таким образом, чтобы не нарушалась детерминированность. Эта идея породила ряд работ, например:

- I-структуры (*англ*. I-structures) [11];
- сети Кана (*англ*. Kahn networks) [17];
- TStreams, Concurrent Collections [13];
- структуры данных на решетках (*англ*. lattice-based data structures) [18, 19].

У этих подходов есть общая черта: переменные, объекты, через которые осуществляется взаимодействие параллельных процессов, меняют свое состояние монотонно вверх на некоторой полурешетке от неопределенного состояния (⊥) к «всё более определенному». При этом верхний элемент решетки (⊤) обозначает «переопределено», «противоречие»; в программе это соответствует ошибке, выработке исключения. Например, множество значений I-структур с целыми числами описывается решеткой, называемой «плоской», состоящей из нижнего элемента «не определено» (⊥), не сравнимых между собой целых чисел и верхнего элемента «переопределено» (⊤). При выполнении операции присваивания значения *y* в переменную со значением *x* в нее записывается наименьшая верхняя грань значений *x* и *y*. Если полученный результат оказывается верхним элементом ⊤, то вырабатывается исключение.

Эта идея в общем виде была проработана в диссертации Lindsey Kuper [18] и в публикациях вместе с ее коллегами [19]. Она доказала детерминированность параллельных вычислений для процессов, взаимодействующих через переменные, принимающие значения из произвольной (полу)решетки.

В нашем проекте мы используем эти идеи, обобщая их на объекты, определяемые пользователем, с монотонно изменяющимся состоянием.

3. Архитектура двухуровневой системы детерминированного параллельного программирования

Поскольку не существует единого набора средств, зафиксированных в языке программирования и библиотеке, обеспечивающих все случаи детерминированного параллельного программирования, система должна быть открытой и входной язык должен позволять как детерминированный, так и недетерминированный код. Но чтобы дать гарантии детерминированности прикладному программисту, входной язык системы должен быть двухуровневым:

- верхний уровень – детерминированная часть: прикладной код на подмножестве языка, который пишет прикладной программист. Ему гарантируется детерминированность любой программы, использующей заготовленные библиотеки нижнего уровня. Принадлежность детерминированному подмножеству проверяется компилятором. Оно примерно соответствует функциональному подмножеству языка Java и ему подобных;
- нижний уровень – недетерминированная часть: библиотеки классов, создаваемые квалифицированными программистами для определенных областей применения на универсальном объектно-ориентированном языке типа Java с богатым набором изобразительных средств, позволяющем кодировать недетерминированные параллельные алгоритмы. Авторы библиотек гарантируют детерминированность параллельных приложений их авторам, программирующим на функциональном подъязыке верхнего уровня.

Такие классы и объекты нижнего уровня мы называем *монотонными*, поскольку оказывается, что, как и в работах, упомянутых в разделе 2.4, их состояние меняется монотонно на некоторой полурешетке, которую, как правило, можно построить по коду класса.

Входным языком такой системы может быть любой объектно-ориентированный язык со средствами параллельного и конкурентного программирования, у которого можно выделить функциональное подмножество. Таковыми являются большинство современных объектно-ориентированных языков. Однако в таком случае некоторые детали эффективной реализации будут «торчать» и загромождать прикладной код. Поэтому мы разрабатываем специализированный Java-подобный язык, названный Ajl [3], в котором эти детали прикрыты синтаксическим сахаром и генерируются из компактного кода. Таким способом мы получаем возможность предлагать пользователям разные механизмы реализации параллельных процессов, тредов, легких тредов (*англ*. light-weight thread).

Мы проводим прототипирование системы, используя платформу JVM[1], языки Java и Kotlin[2] [9], сопрограммы языка Kotlin[3], библиотеку Quasar[4], реализующую легкие потоки («файберы» и «стренды», *англ.* fibers and strand) на JVM. Отслеживаем развитие проекта Loom[5], идущего на смену Quasar с целью более тесной интеграции с языком Java и последующего включения в стандартную Java-библиотеку. Разрабатываем свой язык Ajl [3].

4. Роль идемпотентности

Помимо детерминированности есть еще одно важное понятие, которое из узко математического превращается во всё чаще упоминаемое ценное свойство параллельных и распределенных программ: *идемпотентность*[6]. Операция, вызов процедуры или функции называется *идемпотентной*, если ее можно выполнить повторно с копией аргументов и она выдаст эквивалентный результат и не породит нового побочного эффекта или породит лишь такой, какой не будет программно заметен из вызвавшего операцию кода. Идемпотентную операцию можно прервать, не завершив, а потом вызвать повторно, досчитать до конца и использовать результат повторного вызова вместо неполученного результата первого. Повторное вычисление завершит создание побочного эффекта, и ни эта, ни другие подсистемы не заметят, что вычисление прерывалось и повторялось. Многократное выполнение идемпотентной операции эквивалентно однократному.

Очевидно, что идемпотентность особенно важна в распределенных системах с большим количеством узлов, будь то узлы суперкомпьютера или вычислительные установки, взаимодействующие через интернет. В таких задачах и системах ненулевая частота отказов узлов и подсистем должна учитываться при их проектировании. Реализация идемпотентных операций, которые можно перевызывать при отказе или по таймауту, – самый простой и естественный способ обеспечения надежности и безотказности приложений.

Например, в интернет-протоколе HTTP[7] некоторые операции, изменяющие состояние, идемпотентны, а именно, PUT и DELETE, а операции POST и PATCH неидемпотентные. Эта разница оговорена в спецификации протокола и должна учитываться разработчиками и серверов, и приложений. Всякий раз, когда можно применить идемпотентную операцию, следует это делать. При использовании неидемпотентных операций программирование обработчиков исключительных ситуация становится более сложным, чем с идемпотентными.

В чисто функциональных языках все вызовы функций идемпотентны, так как не создают побочного эффекта. Естественно перенести это свойство и на детерминированное объектно-ориентированное программирование с побочными эффектами. Более того, оказывается, что теория и приемы разработки монотонных объектов становятся более ясными, если проектировать объекты нижнего уровня сразу с учетом этого и детерминированными, и идемпотентными. По нашему опыту разработки монотонных классов, легче позаботиться об идемпотентности, после чего детерминированность часто получается сама собой или нужно приложить лишь немного дополнительных усилий, чтобы ее достичь.

[1] JVM = Java Virtual Machine.
[2] http://kotlin.jetbrains.org/
[3] https://kotlinlang.org/docs/reference/coroutines/coroutines-guide.html
[4] https://github.com/puniverse/quasar
[5] https://wiki.openjdk.java.net/display/loom/Main
[6] https://en.wikipedia.org/wiki/Idempotence
[7] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

5. Понятие монотонного класса и объекта

Резюмируем определение монотонности классов и объектов, как оно используется в нашей разработке и было дано в предыдущих работах [15, 8, 6, 10]. Оно дается не через наличие полурешетки, как в [18, 19], а операционно.

Классы и их объекты называются *монотонными*, если при их использовании в любой программе на функциональном подъязыке все выражения удовлетворяют следующим двум свойствам:

- *Детерминированность*: результаты, полученные в любом порядке параллельных вычислений копий выражения от одного и того же начального состояния, эквивалентны.
- *Идемпотентность*: повторное вычисление копии выражения от состояния, полученного при предыдущем вычислении или параллельно с ним, дает эквивалентный результат и побочный эффект, не отличимый программно на подъязыке верхнего уровня от побочного эффекта первого вычисления.

Эти свойства должны выполняться одновременно для всех монотонных классов, когда они используются вместе, в любой функциональной программе. Здесь используется понятие эквивалентности результатов как их неотличимости средствами языка верхнего уровня.

Обратим внимание на следующую тонкость этого определения: монотонность классов и объектов определяется не через их свойства как таковые, а через свойства *любой* функциональной программы, использующей эти классы. Поэтому доказательства монотонности, вообще говоря, нетривиальны.

В настоящее время мы конструируем монотонные классы, рассуждая об этих свойствах неформально и наивно. В будущем мы рассчитываем, что удастся разработать средства автоматизации доказательства монотонности в подавляющем числе случаев, хотя, конечно, полностью автоматических компьютерных инструментов на все случаи жизни никогда построено не будет, как и в общем случае средств верификации программ.

6. Пример монотонного класса и его использования

Приведем пример простого монотонного класса, имеющего практическое значение, который был изобретен более 30 лет назад в связи с разработкой архитектур с управлением потоком данных (*англ.* dataflow computers). Речь об I-структурах [11].

I-структурой называется переменная целочисленного типа (или другого примитивного, то есть нессылочного) со следующей семантикой операций:

- при создании переменная находится в состоянии «не определено», «не готово»;
- при чтении:
  - если переменная имеет состояние «не готово», процесс, вызвавший эту операцию, переходит в состояние ожидание записи в эту переменную каким-либо процессом и будет продолжен после записи;
  - в противном случае выдается текущее значение переменной;
- при записи значения x:
  - если состояние переменной «не готово», x записывается в нее и она переходит в состояние «готово»;
  - если значение переменной было равно x, то ничего не делается;
  - в противном случае вырабатывается исключение.

```
class LongM {

  private boolean unready = true;
  private long value;

  public synchronized long get() {

    if (unready) wait();
    return value;
  }

  public synchronized void set(long x) {

    if (unready) {
      value = x;
      unready = false;
      notifyAll();
    }
    else if (x != value)
      throw new RuntimeException();
  }
}
```

Рис. 1. Монотонный класс `LongM` для хранения значения типа `long` на Java

```
class LongM {

  private var unready = Suspender()
  private var value: Long = 0;

  suspend fun get(): Long {

     unready?.suspend()
     return value
  }

  fun set(x: Long) {
    val u = unready
    if (u != null) {
      value = x
      unready = null
      u.resumeAll()
    }
    else if (x != value)
      throw Exception()
  }
}
```

Рис. 2. Монотонный класс `LongM` для хранения значения типа `Long` на Kotlin

```
suspend fun fib(n: Int): Long = coroutineScope {

    val v = LongM()
    val w = LongM()
    fibRec(n, v, w)
    w.get()
}

suspend fun fibRec(n: Int, v: LongM, w: LongM): Unit = coroutineScope {

    if (n <= 1) {
        v.set(1)
        w.set(1)
    }
    else {
        val u = LongM()
        launch { w.set(u.get() + v.get()) }
        fibRec(n - 1, u, v)
    }
}
```

Рис. 3. Пример программы на языке верхнего уровня – функциональном подмножестве языка Kotlin, использующем механизм сопрограмм и их параллельного выполнения: вычисление чисел Фибоначчи `fib(n)` с запоминанием промежуточных значений в монотонных объектах `LongM`

На рис. 1 и 2 изображен код на языках Java и Kotlin класса `LongM` с операциями чтения `get` и записи `set` целых чисел типа `long` с семантикой I-структуры. На рис. 1 класс `LongM` реализован с использованием стандартных средств многопоточного программирования языка Java. Булевская переменная `unready` указывает, определено ли значение поля `value`, в которое записывается аргумент операции `set`.

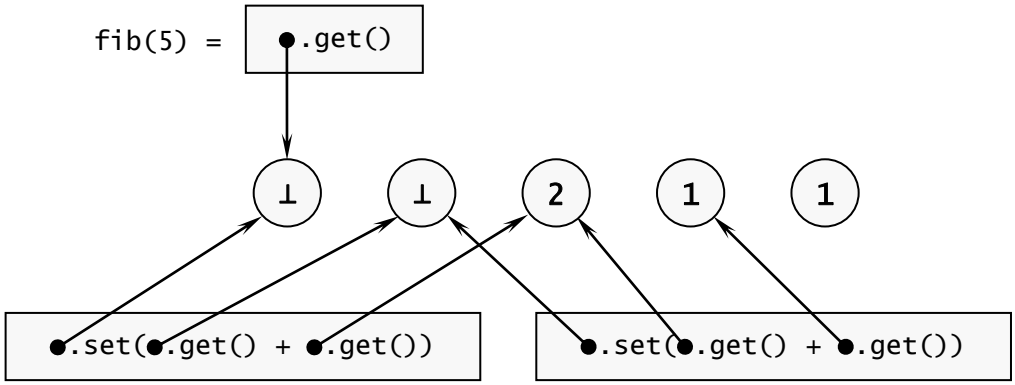


Рис. 4. Состояние вычисления `fib(5)`, после того как рекурсивная функция `fibRec` отработала, но оставила незавершившиеся сопрограммы, созданные операторами `launch`. Часть из них выполнилась и заполнила три монотонных объекта `LongM` значениями `2 = fib(2)`, `1 = fib(1)`, `1 = fib(0)`. Две другие изображены прямоугольниками с выражениями `u.set(v.get() + w.get())` со ссылками на объекты на местах переменных `u`, `v`, `w`. Объекты класса `LongM` изображены кругами, где знак ⊥ обозначает «неготовое» состояние, число – значение поля `value`.

Рис. 2 содержит код класса с такой же семантикой на языке Kotlin с использованием его средств параллельного программирования на сопрограммах. Методы, помеченные ключевым словом **`suspend`**, имеют возможность «задерживаться» некоторыми операциями. Стеки вызовов таких методов называются в языке Kotlin *сопрограммами*. Здесь метод `get` задерживается операцией `unready?.suspend()` до тех пор, пока переменная `value` не станет «готовой». Объекты вспомогательного класса `Suspender` поддерживают очередь задержанных сопрограмм. Метод `suspend` добавляет текущую сопрограмму в очередь, после чего управление передается диспетчеру, возобновляющему выполнение другой готовой сопрограммы. Метод `resumeAll` освобождает все сопрограммы в данной очереди, то есть передает их диспетчеру, который их продолжит, когда у него будет такая возможность.

Мы утверждаем (без доказательства), что все программы на функциональном подмножестве языка Java или Kotlin, использующие класс `LongM`, обладают детерминированностью и идемпотентностью, то есть класс `LongM` является монотонным.

По сравнению с реализацией класса `LongM` на Java на «тяжелых» тредах сопрограммы языка Kotlin – это «легкое» средство параллельного программирования. Кроме того, параллельные программы на языке Kotlin более компактны и читабельны, чем на Java. Поэтому пример использования класса `LongM` приведем на рис. 3 на языке Kotlin. В коде встречаются два идентификатора, относящихся к параллельному программированию: метод `coroutineScope`, употребление которого воспринимайте как шаблон программирования сопрограмм, запускающих другие сопрограммы, и метод `launch`, создающий сопрограмму с выражением из аргумента в качестве ее начального состояния. Эта сопрограмма передается диспетчеру и попадает в очередь готовых, а вычисление продолжается после `launch`. Отметим, что оператор создания объекта записывается в языке Kotlin без ключевого слова **`new`**.

Программа на рис. 3 вычисляет числа Фибоначчи за линейное время благодаря сохранению промежуточных значений в объектах `LongM`. Функция `fib(n)` – головная, `fibRec(n, v, w)` – вспомогательная. Ее последний аргумент `w` – монотонный объект класса `LongM` по окончании вычисления принимает значение числа Фибоначчи `fib(n)`. Рис. 4 иллюстрирует ход вычисления `fib(5)`, показывая некоторое промежуточное состояние сопрограмм и объектов.

Заметим, что код на рис. 4 напоминает программирование на языке Prolog. Это неудивительно: логические переменные языка Prolog – это пример монотонных объектов. Однако

наша задача не ограничивается воспроизведением элементов логического программирования на объектно-ориентированном языке, а направлена на предоставление более универсальных и эффективных средств работы с объектами с явным использованием ссылок, чего нет ни в чисто функциональном, ни в логическом программировании.

7. Примеры задач, решаемых с монотонными объектами

Для изучения и демонстрации возможностей программирования с монотонными объектами мы разрабатываем библиотеку монотонных классов на следующих классах задач.

7.1. Порождение и обработка графов

Классические чисто функциональные языки не дают возможности обрабатывать графы эффективно, так, чтобы узлы были представлены объектами, а связи между ними – ссылками в полях объектов. В функциональных языках можно эффективно представлять только деревья. В системе программирования с монотонными объектами мы сохраняем большинство положительных свойств функциональных языков и расширяем область эффективно представимых данных с деревьев до произвольных графов.

Однако создание столь же эффективного представления графа средствами ограниченных побочных эффектов, какие допустимы для монотонных объектов, – нетривиальная задача. Посмотрим, что будет, если в классе `LongM` на рис. 1 или 2 заменить тип переменной `value` с примитивного `long` (`Long` соответственно) на ссылочный `Object`, переименовав класс в `ObjectM`, чтобы название отражало смысл. Тогда класс станет немонотонным! Чтобы показать это, рассмотрим такую последовательность предложений (в синтаксисе языка Kotlin):

```
val a: ObjectM = ObjectM()
a.set(ObjectM())                // (1) первое вычисления выражения
a.set(ObjectM())                // (2) второе вычисление такого же выражения
```

Строки (1) и (2) содержат одинаковые выражения, а идемпотентность требует, чтобы выражение (2) вычислялось с таким же результатом, что и выражение (1), и без нового побочного эффекта. Однако после выполнения (1) полю `a.value` будет присвоена ссылка на объект, созданный на строке (1), а вызов `set` в строке (2) попытается записать в то же поле неравную ссылку на новый объект `ObjectM()`, что вызовет выработку исключения.

В работе [10] эта проблема разбирается подробнее и предлагается несколько решений для записи ссылок в монотонные объекты, два из которых позволяют порождать циклические структуры данных, столь же эффективные, как и при обычном, не ограниченном монотонностью, объектно-ориентированном программировании.

Сформулируем идею одного из них, чтобы показать, что описанные трудности преодолимы. Заведем в данном классе, претендующем на монотонность (похожем на `ObjectM` или другом), «фабрику» объектов, которая создает не один объект, а сразу массив или список из указанного числа новых объектов. Этот список не только выдается в качестве результата фабрики, но и сохраняется в приватном поле каждого из созданных объектов. У данного монотонного класса могут быть ссылочные поля. Определим операцию `set` так, чтобы она позволяла записать в эти поля только ссылки из сохраненного списка, то есть на объекты, созданные *одновременно* с данным объектом. При обращении к `set` со ссылкой, отсутствующей в списке, выдается исключение. Можно убедиться, что классы и объекты с такой операцией записи `set` и фабрикой одновременного создания объектов являются монотонными.

Недостаток такого решения: при создании представления графа надо заранее знать число узлов в нем, чтобы одновременно породить нужное число представляющих их объектов. Это «плата» за детерминированность и идемпотентность.

7.2. Переборные алгоритмы типа метода ветвей и границ

С монотонными объектами можно запрограммировать, конечно, не все задачи параллельного программирования, а только некоторую часть тех, у которых результат однозначен. Именно таковы оптимизационные задачи типа поиска кратчайшего пути в графе методом ветвей и границ (*англ*. branch-and-bound).

По смыслу задачи в них присутствуют несколько видов монотонно изменяемых данных: монотонно меняется рекорд (кратчайший путь, найденный к настоящему времени), причем порядок его изменения зависит от порядка исполнения параллельных процессов, но результат не зависит; монотонно растут пути, причем их много. Кроме того, необходима эквивалентность реализаций с полным перебором и с «отсевом» (*англ*. pruning) вариантов путей, которые заведомо не улучшат рекорд. Также желательно, чтобы «переключение» реализации с полным перебором на вариант с отсевом делалось в одном месте кода, чтобы упростить доказательство их эквивалентности. Гарантии эквивалентности этих вариантов и монотонности должны быть реализованы в как можно меньшем по размеру коде монотонных классов.

С точки зрения методов разработки монотонных классов эта задача интересна тем, что для реализации «отсева» нужны монотонные объекты «высшего порядка», операции которых принимают в качестве аргумента лямбда-выражение, описывающее продолжения вычислений, которые могут подвергнуться «отсеву».

Эта задача также оказалась нетривиальной и поучительной. Подробная информация о методике программирования таких задач будет опубликована в наших будущих работах.

8. Заключение

В статье дана мотивировка и принципы построения системы параллельного объектно-ориентированного программирования, разрабатываемой авторами на основе понятия монотонных классов и объектов. Ее основная идея – двухуровневый входной язык, состоящий из универсального объектно-ориентированного языка «нижнего» уровня (типа Java, Kotlin и т.п.), предназначенного для экспертов в параллельном программировании, создающих библиотеки, и языка «верхнего» уровня – подмножества, близкого к функциональному языку, предназначенного для прикладных программистов. Дано операционное определение понятия монотонности классов и объектов как таких, которые гарантируют использующим их программам на подъязыке верхнего уровня детерминированность и идемпотентность.

Для демонстрации реализуемости и практичности данного подхода приведен пример на языке Kotlin для монотонного класса, реализующего понятие I-структуры, и функции Фибоначчи, его использующей. Пример показывает, что язык Kotlin обладает уже достаточным набором понятий, чтобы развивать на нем методику, которую можно назвать *монотонным программированием*. Однако в общем случае многие понятия реализации будут «торчать» в прикладном коде. Поэтому целесообразна разработка Java/Kotlin-подобного языка, в котором эти элементы реализации «прикрыты» высокоуровневыми понятиями. Кроме того, его компилятор будет проверять принадлежность прикладного кода языку, на котором гарантируется детерминированность и идемпотентность. Мы ведем разработку такого языка с рабочим названием Ajl, и его завершение является нашей будущей работой.

Центральная часть данного проекта – разработка библиотек монотонных классов для различных областей применения, гораздо более сложных, чем приведенный пример. Были охарактеризованы два типа прикладных задач, на которых система сейчас отрабатывается и будет продемонстрирована в будущих публикациях. Не существует конечного набора классов, которые удовлетворили бы все потребности монотонного параллельного программирования, поэтому эта библиотека открыта и должна развиваться и пополняться. Главное предназначение разрабатываемой системы – сделать этот процесс легким и практичным.

Литература

1. *Абрамов С. М., Адамович А. И., Коваленко М. Р.* Т-система – среда программирования с поддержкой автоматического динамического распараллеливания программ. Пример реализации алгоритма построения изображений методом трассировки лучей // Программирование. 1999. Т. 25, № 2. С. 100–107.
2. *Адамович А. И.* Струи как основа реализации понятия Т-процесса для платформы JVM // Программные системы: теория и приложения. 2015. Т. 6, № 4. С. 177–195. DOI: 10.25209/2079-3316-2017-8-4-221-244.
3. *Адамович А. И.* Язык программирования Ajl: автоматическое динамическое распараллеливание для платформы JVM // Программные системы: теория и приложения. 2016. Т. 7, № 4. С. 83–117. DOI: 10.25209/2079-3316-2016-7-4-83-117.
4. *Адамович А. И., Климов А. В.* Об опыте использования среды метапрограммирования Eclipse/TMF для конструирования специализированных языков // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции. М.: ИПМ им. М. В. Келдыша РАН, 2016. С. 3–8. DOI: 10.20948/abrau-2016-45.
5. *Адамович А. И., Климов А. В.* Как создавать параллельные программы, детерминированные по построению? Постановка проблемы и обзор работ // Программные системы: теория и приложения. 2017. Т. 8, № 4. С. 221–244. DOI: 10.25209/2079-3316-2017-8-4-221-244.
6. *Адамович А. И., Климов А.В*. Принципы построения системы детерминированного параллельного программирования // Наукоемкое программное обеспечение: труды семинара 12-й Междунар. Ершовской конф. по информатике (ПСИ’19). 2–3 июля 2019 г., Новосибирск. С. 26–33. ISBN 978-5-4437-0909-3.
7. *Андрианов А. Н., Баранова Т. П., Бугеря А. Б., Ефимкин К. Н.* Трансляция непроцедурного языка Норма для графических процессоров // Препринты ИПМ им. М. В. Келдыша. 2016. № 73. 24 с. DOI: 10.20948/prepr-2016-73.
8. *Климов А. В.* Детерминированные параллельные вычисления с монотонными объектами // Научный сервис в сети Интернет: многоядерный компьютерный мир: труды Всероссийской научной конференции. М.: Изд-во Московского университета, 2007. С. 212–217.
9. *Скин Д., Гринхол Д*. Kotlin. Программирование для профессионалов. СПб.: Питер, 2020.
10. *Adamovich A.I., Klimov A.V.* Building Cyclic Data in a Functional-Like Language Extended with Monotonic Objects // X Workshop PSSV: Program Semantics, Specification and Verification: Theory and Applications. Abstracts. 2019. P. 11-19. ISBN 978-5-4437-0918-5.
11. *Arvind, Nikhil R. S., Pingali K. K.* I-structures: Data Structures for Parallel Computing // ACM Trans. Program. Lang. Syst. 1989. V. 11, № 4. P. 598–632. DOI: 10.1145/69558.69562.
12. *Bocchino R. L. (Jr.), Adve V. S., Adve S. V., Snir M.* Parallel Programming Must Be Deterministic by Default // Fifth USENIX Conference on Hot Topics in Parallelism, HotPar’09. USENIX Association, 2009.
13. *Burke M. G., Knobe K., Newton R., Sarkar V.* Concurrent Collections Programming Model // Encyclopedia of Parallel Computing / ed. D. Padua. Springer US, 2011. P. 364–371. DOI: 10.1007/978-0-387-09766-4_238.
14. *Hammond K., Michelson G.* (eds.). Research Directions in Parallel Functional Programming, London, UK: Springer, 1999.
15. *Klimov A. V.* Dynamic Specialization in Extended Functional Language with Monotone Objects // SIGPLAN Not. 1991. V. 26, № 9. P. 199–210. DOI: 10.1145/115865.376287.
16. *Potanin A., Östlund J., Zibin Y., Ernst M. D.* Immutability // Aliasing in Object-Oriented Programming / eds. D. Clarke, J. Noble, T. Wrigstad. LNCS 7850. Springer, 2013. P. 233–269.
17. *Kahn G.* The Semantics of a Simple Language for Parallel Programming // IFIP Congress, 1974. P. 471–475.

18. *Kuper L.* Lattice-based Data Structures for Deterministic Parallel and Distributed Programming, Ph.D. Thesis. IN, USA: Indiana University, 2015. 253 p.
19. *Kuper L., Todd A., Tobin-Hochstadt S., Newton R. R.* Taming the Parallel Effect Zoo: Extensible Deterministic Parallelism with LVish // ACM SIGPLAN Not. 2014. V. 49, № 6. P. 2–14. DOI: 10.1145/2666356.2594312.
20. *Marlow S.* Parallel and Concurrent Programming in Haskell. CA, USA: O'Reilly, 2013.

Статья поступила в редакцию 31.07.2019;
переработанный вариант – 02.09.2019.

Адамович Алексей Игоревич
старший научный сотрудник Исследовательского центра мультипроцессорных систем Института программных систем им. А. К. Айламазяна РАН (152021, Ярославская обл., Переславский район, с. Веськово, ул. Петра Первого, д. 4 «а»), тел. (4852) 695-228, e-mail: lexa@adam.botik.ru.

Климов Андрей Валентинович
старший научный сотрудник отдела «Программное обеспечение высокопроизводительных вычислительных систем и сетей» Института прикладной математики им. М. В. Келдыша РАН (125047, Москва, Миусская пл., д. 4), тел. (4852) 695-228, e-mail: klimov@keldysh.ru.

An approach to deterministic parallel programming system construction based on monotonic objects

A. I. Adamovich, A. V. Klimov

Due to the explosive growth of the complexity of programs for multi-core processors and supercomputers, the idea of parallel computation with determinism guaranteed by the programming language and system is becoming increasingly significant. This paper analyzes the problem of making parallel programming as deterministic as possible and some of the existing approaches to its solution. It describes the principles of constructing the object-oriented programming system developed by the authors, which allows expert programmers to write both deterministic and non-deterministic code with guarantees to application developers that their programs are deterministic. The system and its input language have two levels: the higher level is intended for application developers; the lower level is for developers of class libraries referred to as *monotonic*. The input language of the higher-level subsystem is like a functional language with the possibility of creating and using immutable and monotonic objects. The libraries of monotonic classes ensure that all programs in the higher-level sublanguage that use only these classes are deterministic and idempotent when they are parallelized by asynchronous calls of all functions. Some representative applications implemented on this system are described.

Keywords: models of parallel computation, deterministic programs, functional programming, object-oriented programming, monotonic objects.

УДК 519.68+681.513.7+612.8.001.57

Временные и пространственные понятия в текстах на естественном языке и их исследование

Т. В. Батура, Л. В. Ефимова, А. С. Еримбетова,
А. Б. Касекеева, Ф. А. Мурзин[1]

Целью работы является создание базы знаний, содержащей информацию о временных и пространственных понятиях, встречающихся в текстах на естественном языке. Основа базы: наиболее важные понятия, относящиеся к времени и пространству, из толкового словаря С. И. Ожегова; перефразированные варианты предложений; результаты анализа словарных статей (толкований) и примеров использования в художественной литературе соответствующих понятий из словаря С. И. Ожегова с помощью программных систем Link Grammar Parser и ДИАЛИНГ и т.д. Результаты работы могут быть использованы в интеллектуальных системах поиска информации. Диаграммы, полученные на выходе систем Link Grammar Parser и ДИАЛИНГ, представляют собой чрезвычайно интересный материал для дальнейших исследований. Целесообразно исследовать возможности применения в компьютерной лингвистике ряда конструкций и понятий математической логики, таких как конструкция Л. Генкина, реализация и опускание типов, модельная полнота, форсинг, а также ряда неклассических логик.

Ключевые слова: компьютерная лингвистика, семантика, временные и пространственные понятия, Link Grammar Parser, ДИАЛИНГ.

1. Введение

Основная цель состоит в том, чтобы исследовать конструкции на естественном языке, содержащие временные и пространственные понятия, в семантическом плане.

Для этого создается база знаний, содержащая информацию о временных и пространственных понятиях. База знаний содержит наиболее важные понятия, относящиеся к времени и пространству, из толкового словаря С. И. Ожегова; перефразированные варианты ряда предложений; результаты анализа словарных статей (толкований) и примеров использования в художественной литературе соответствующих понятий из словаря С. И. Ожегова с помощью программных систем Link Grammar Parser и ДИАЛИНГ.

Далее предполагается рассмотреть диаграммы, полученные на выходе систем Link Grammar Parser и ДИАЛИНГ средствами математической логики. Речь идет об исследовании возможностей применения в компьютерной лингвистике ряда конструкций и понятий математической логики, таких как конструкция Л. Генкина, реализация и опускание типов, модельная полнота, форсинг, а также ряда неклассических логик.

Предполагается провести анализ средствами математической логики свойств лексических функций И. А. Мельчука в контексте определения смысла текста и теоретико-множественных моделей языка, предложенных С. Маркусом.

[1] Исследования выполнены при финансовой поддержке Министерства образования и науки Республики Казахстан (грант 2018-2020 MES RK № AP05133550), Интеграционного проекта СО РАН (АААА-А18-118022190008-8) и Российского фонда фундаментальных исследований (грант № 19-07-01134).

В будущем работа будет также включать исследование комбинаторных свойств лингвистических определений и конструкций, возможностей применения в компьютерной лингвистике понятий из теоретико-множественной и алгебраической топологии и исследование географических понятий.

Результаты работы могут быть использованы в интеллектуальных системах поиска информации. А именно, для определения релевантности текста поисковому запросу и для определения тем текстов. Она также представляет интерес для ученых-лингвистов.

2. Формальные методы исследования семантики текстов

Семантика – раздел лингвистики, изучающий смысловое значение единиц языка: отдельных слов, словосочетаний, предложений, фрагментов текста. На данный момент существует ряд машинно-ориентированных методов отображения смысла высказываний.

Например, И. А. Мельчук ввел понятие лексической функции, развил понятия синтаксических и семантических валентностей и рассмотрел их в контексте толково-комбинаторного словаря [1]. В. Ш. Рубашкин и Д. Г. Лахути ввели иерархию синтаксических связей для более эффективной работы семантического анализатора. Подход И. А. Мельчука поддержан в программной системе ДИАЛИНГ [2].

Появилось понятие универсального языка представления знаний. Он может быть удобным инструментом для осуществления вывода новых знаний из уже имеющихся. Вполне возможно, что именно в направлении создания подобных семантических языков будут развиваться исследования в будущем. Например, в настоящее время система Knowledge Vault содержит 1.6 миллиарда фактов. Система NELL, разрабатываемая в рамках проекта ReadTheWeb университетом Карнеги – Меллона, содержит более 50 миллионов утверждений, дополнительно характеризующихся различными степенями доверия.

Еще один подход – это использование синтаксического анализатора Link Grammar Parser [3], разработанного в университете Карнеги – Меллона, базирующегося на некоторой специальной теории синтаксиса. Отметим, что данная теория, вообще говоря, отличается от классической теории синтаксиса. Получив предложение, система приписывает к нему синтаксическую структуру, которая состоит из множества помеченных связей (коннекторов), соединяющих пары слов.

Получаемые диаграммы, по сути, являются аналогами так называемых деревьев подчинения предложений. В деревьях подчинения от главного слова в предложении можно задать вопрос к второстепенному. Таким образом, слова выстраиваются в древовидную структуру. Главной причиной, по которой анализатор называют семантической системой, можно считать уникальный по полноте набор связей. Имеется около 100 основных связей, при этом некоторые из них дополнительно имеют 3–4 варианта.

Наши исследования проводятся в соответствии со следующими пунктами.

1. Выборка наиболее важных понятий, относящихся к времени и пространству, из толкового словаря С. И. Ожегова.

2. Создание массива перефразированных вариантов различных предложений и методов оценивания их похожести.

3. Анализ словарных статей (толкований) и примеров использования в художественной литературе соответствующих понятий из словаря С. И. Ожегова с помощью программных систем Link Grammar Parser и ДИАЛИНГ.

4. Анализ диаграмм, полученных на выходе систем Link Grammar Parser и ДИАЛИНГ средствами математической логики. Исследование возможностей применения к географическим понятиям.

5. Интеграция результатов исследований в базу знаний.

3. Программная система Link Grammar Parser

Link Grammar Parser – это синтаксический анализатор английского языка, разработанный в университете Карнеги – Меллона. Как уже было отмечено выше, она базируется на неклассической теории синтаксиса. Link Grammar Parser имеет словари, включающие около 60 000 словарных форм. Он позволяет анализировать большое число синтаксических конструкций, включая многочисленные редкие выражения и идиомы. Анализатор довольно устойчив; может пропустить часть предложения, непонятную ему, и определить структуру оставшейся части предложения. Он способен делать разумные предположения о синтаксической категории неизвестных ему слов (т.е. слов, которые отсутствуют в словарях) из контекста и написания. У анализатора есть данные об именах собственных, о числовых выражениях и разнообразных знаках препинания.

Для каждого слова в словаре записывается, какими коннекторами оно может быть связано с другими словами предложения. Коннектор состоит из имени типа связи, в которую может вступать рассматриваемая единица анализа. Например, пометка S соответствует связи между субъектом и предикатом, O – между объектом и предикатом. Только основных, наиболее важных связей, имеется более ста. Для обозначения направления связи справа к коннектору присоединяется знак «+», слева – знак «–». Левонаправленный и правонаправленный коннекторы одного типа образуют связь (link).

Например, если слову W1 приписан коннектор A+, а слову W2 – коннектор A–, то в синтаксической структуре предложения, состоящего из двух слов W1 W2, будет проведена связь A между словами W1 и W2. Предложение же W2 W1 не получит никакой интерпретации, поскольку W2 приписан коннектор A–, который образует связь только влево, а слову W1 приписан A+, который образует связь только вправо. Отметим, что может быть несколько вариантов разбора одного и того же предложения. На рис. 1 приведены 4 варианта разбора предложения «Вблизи ветер дул сильнее».

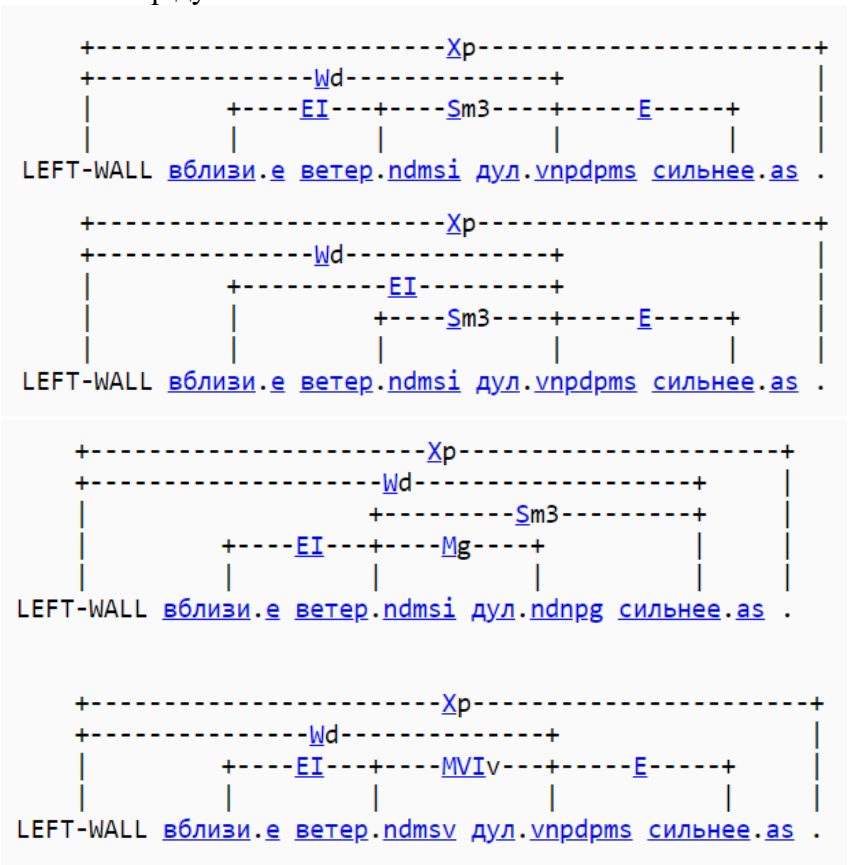


Рис. 1. Варианты разбора предложения при помощи Link Grammar Parser

Здесь приняты следующие обозначения:

Xp – связь для соединения начала предложения с точкой (в конце предложения).

Wd – связь указывает на вершину предложения.

EI – связь с наречием, обозначающим пространственное понятие.

Sm3 – соединяет подлежащее с глаголом (строчные буквы дополняют информацию о роде и лице).

E – связь между глаголом и наречием, модифицирующим его.

Mg – связь между существительным и причастием.

MVIv – соединяет сказуемое с дополнением.

Мы видим, что связь Xp присутствует во всех схемах. Можно сказать, что она инвариантна. Слово *вблизи*, относящееся к пространственным понятиям, считаем константой в нашей сигнатуре. Относительно других не инвариантных связей с помощью логических формул можно записать имеющиеся эквивалентности.

$$x_1 = \textit{ветер},$$
$$x_2 = \textit{дул},$$
$$\varphi_1 = EI(\textit{вблизи}, x_1) \wedge Sm3(x_1, x_2),$$
$$\varphi_2 = EI(\textit{вблизи}, x_2) \wedge Sm3(x_1, x_2),$$
$$\varphi_3 = EI(\textit{вблизи}, x_1) \wedge Mg(x_1, x_2),$$
$$\varphi_4 = EI(\textit{вблизи}, x_1) \wedge MVIv(x_1, x_2),$$
$$\varphi_1 \leftrightarrow \varphi_2 \leftrightarrow \varphi_3 \leftrightarrow \varphi_4.$$

4. Перефразирования предложений

Известно, что естественные языки обладают многообразием способов выражения смысла, одну и ту же мысль можно передать разными словами. Эта особенность значительно затрудняет анализ текстов на естественном языке, так как задача обнаружения похожего смысла в разных высказываниях является трудно формализуемой. Формализация анализа временных и пространственных понятий в текстах позволяет хотя бы частично приблизиться к решению этой задачи. В данном разделе формально описан процесс обнаружения перефразированных предложений, содержащих некоторые пространственные и временные понятия.

В ряде случаев в базе данных целесообразно иметь перефразированные варианты предложений и методы оценивания их близости или, иначе говоря, похожести. Это особенно важно для предложений, содержащих глаголы каузации движения и изменения положения в пространстве, ввиду сложности предложений, их содержащих. В этот класс входят глаголы, выражающие такие понятия, как: перемещать, приближать, удалять, нести, давать, брать, класть, поднимать, опускать, бросать, ловить, посылать и др. Определенный интерес представляют предложения, содержащие некоторые наречия: впереди, позади, сбоку, раньше, позже, еще, уже; именно с точки зрения их перефразирования.

В [4] дано описание, каким образом можно сравнивать перефразированные предложения, для случая использования анализатора Link Grammar Parser. Предположим, что L – множество слов некоторого естественного языка. Для любого слова $x \in L$ обозначим $Norm(x)$ его нормализованную форму. Запись $Syn(x, y)$ обозначает, что x, y – синонимы.

Возникают два вида эквивалентностей:

1) $x_1 \approx x_2 \leftrightarrow x_1 = x_2 \vee Syn(x_1, x_2)$;

2) $x_1 \equiv x_2 \leftrightarrow Norm(x_1) = Norm(x_2)$.

Предложение рассматриваем как вектор с компонентами из слов $\bar{x} =< x_1,\ldots,x_n >$. Функция $Norm$ может быть естественно распространена на предложения $Norm(\bar{x}) =< Norm(x_1),\ldots,Norm(x_n) >$. Текст $T = < \bar{x}_1,\ldots,\bar{x}_n >$ есть последовательность предложений.

Пусть запись $\bar{x} \models P(x_i, x_j)$ обозначает, что в схеме разбора предложения $\bar{x} =< x_1,\ldots,x_n >$ посредством анализатора Link Grammar Parser имеется коннектор типа P, идущий от слова x_i к слову x_j. Знак $\models$ означает, что фактически мы имеем дело с моделью. Основным множеством модели является множество пар $\{<1, x_1 >,\ldots,< n, x_n >\}$. Так как одно и то же слово может входить в предложение два и более раз, то это приводит к необходимости рассмотрения именно пар, а не отдельных слов. В силу сказанного выше корректным является даже обозначение $\bar{x} \models \varphi$, где φ – формула, например, исчисления предикатов первого порядка. Фактически $\bar{x}$ одновременно является обозначением и для вектора, и для модели.

Предположим, что даны два предложения $\bar{x} = < x_1,\ldots,x_n >$, $\bar{y} =< y_1,\ldots,y_m >$. Интерес представляют функции f, такие, что $dom(f) \subseteq \{1,\ldots,n\}$, $range(f) \subseteq \{1,\ldots,m\}$ с дополнительными свойствами типа: $f(i) = j \to x_i \approx y_j$, $f(i) = j \to x_i \equiv y_j$ и другие подобные им.

При сопоставлении двух предложений, точнее при анализе их на близость, осуществляется проверка ряда логических свойств. Например, пусть $f(i_1) = j_1$, $f(i_2) = j_2$. Теперь приведем примеры такого рода свойств.

Инвариантность коннектора:

$\bar{x} \models P(x_{i_1}, x_{i_2}) \to \bar{y} \models P(y_{j_1}, y_{j_2})$.

Замена коннектора на дизъюнкцию других:

$\bar{x} \models P(x_{i_1}, x_{i_2}) \to \bar{y} \models \bigvee_t Q_t(y_{j_1}, y_{j_2})$.

Расщепление коннектора на два коннектора:

$\bar{x} \models P(x_{i_1}, x_{i_2}) \to \exists k\, (\bar{y} \models Q(y_{j_1}, y_k) \wedge R(y_k, y_{j_2}))$.

Расщепление коннектора на два коннектора с инверсией:

$\bar{x} \models P(x_{i_1}, x_{i_2}) \to \exists k\, (\bar{y} \models Q(y_{j_2}, y_k) \wedge R(y_k, y_{j_1}))$.

Принимая во внимание, что $\bar{y}$ является обозначением для соответствующей модели, формула из третьего пункта может быть переписана в виде $\bar{x} \models P(x_{i_1}, x_{i_2}) \to \bar{y} \models \exists y Q(y_{j_1}, y) \wedge R(y, y_{j_2})$. По аналогии может быть записана формула из четвертого пункта.

Резюмируя, можно сказать, что в нашем распоряжении имеются правила вида

$R_i : \bar{x} \models \varphi_i(x_1, x_2) \to \bar{y} \models \psi_i(y_1, y_2)$.

Отметим, что для английского языка такого рода правил нами зафиксировано более тридцати. Для русского и других языков этот вопрос менее изучен. Для системы ДИАЛИНГ тоже могут быть сформулированы аналогичные правила, но это более сложный вопрос.

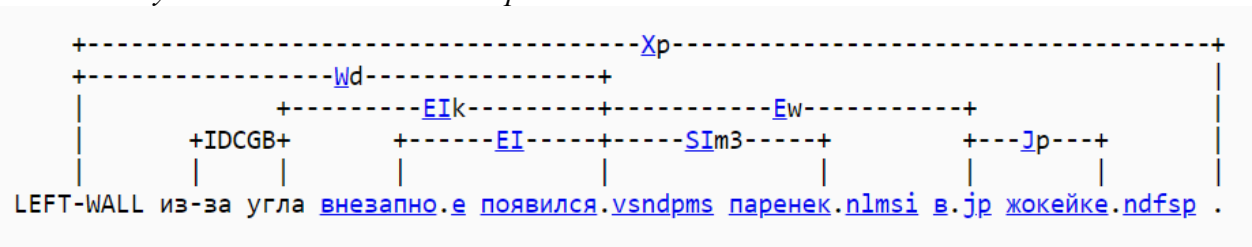
Далее строится функция f и проводится анализ по поводу выяснения того, встречаются ли индексы i_1, i_2, $j_1 = f(i_1)$, $j_2 = f(i_2)$, такие, что на конкретных словах из предложений $\bar{x}, \bar{y}$ выполнено правило R_i, т.е. $\bar{x} \models \varphi_i(x_{i_1}, x_{i_2}) \to \bar{y} \models \psi_i(y_{j_1}, y_{j_2})$. Для простоты можно говорить, что правило выполняется на паре $< i_1, i_2 >$.

Теперь рассмотрим множество всех таких пар $< i_1, i_2 >$, на которых выполнено одно из правил. Обозначим это множество I, и пусть его мощность $|I| = n$. Отметим, что анализатор Link Grammar Parser допускает между двумя словами наличие только одного коннектора. Поэтому будет выполняться не более чем одно правило.

Далее пусть n_1, n_2 – количество коннекторов, получающихся в результате анализа предложений $\bar{x}, \bar{y}$ соответственно. В качестве меры похожести двух предложений можно ввести $\mu_0(\bar{x}, \bar{y}) = n / \max(n_1, n_2)$ или $\mu_1(\bar{x}, \bar{y}) = 2n/(n_1 + n_2)$.

Известно, что предлог – это служебная часть речи, выражающая синтаксические отношения между именем существительным, местоимением, числительным и словами других частей речи, а также между существительными. Часть предлогов совмещают ряд значений. Так, предлоги *за, под, из, от, в, на* совмещают причинные, пространственные и временные значения. Предлог *через*, выражая пространственные (*через горы*) и временные (*через века*) отношения, в просторечии встречается при выражении причинных отношений (*через тебя я лишился семьи*). Другие предлоги совмещают причинные значения со значениями цели (например, *для, по*). На рис. 2 приведен пример перефразирования довольно простого предложения, содержащего предлог *из-за*.

1. *Из-за угла внезапно появился паренёк в жокейке*.



2. *Внезапно появившийся из-за угла паренёк был в жокейке*.

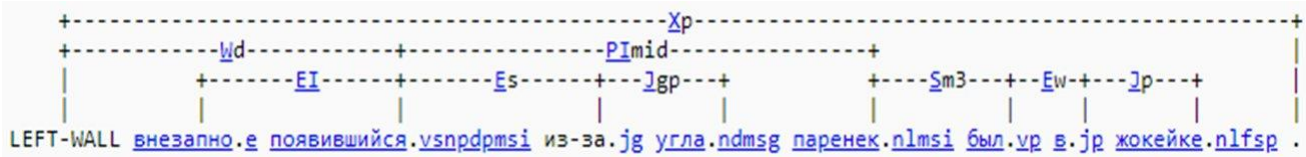


Рис. 2. Пример перефразирования предложения с предлогом *из-за*

Здесь приняты следующие обозначения в дополнение к тем, которые имеются на рис. 1:

Ew – соединяет глагол с предлогом.

J – соединяет предлог с существительным (строчные буквы дополняют информацию о падеже).

PImid – связь существительного с пассивным причастием прошедшего времени.

Очевидно, что здесь имеются инвариантные коннекторы, например: Xp, Jp, EI. Таковым можно считать даже Wd. Слова *появился*. *появившийся* считаем эквивалентными. Далее, слова *из-за, внезапно, появился*, относящиеся к пространственным и временным понятиям, полагаем константами в нашей сигнатуре. Логические формулы, приведенные ниже, позволяют описать перефразирования.

$$x_1 = угол,$$

$$x_2 = паренёк,$$

$$\varphi_{11} = IDBCP(из-за, x_1) \wedge EIk(x_1, появился) \wedge EI(внезапно, появился),$$

$$\varphi_{12} = S\,\mathrm{Im}3(появился, x_2),$$

$$\varphi_1 = \varphi_{11} \wedge \varphi_{12},$$

$$\varphi_{21} = EI(внезапно, появившийся) \wedge Es(появившийся, из-за) \wedge Jgp(из-за, x_1),$$

$$\varphi_{22} = \mathrm{P}\,\mathrm{Im}id(появившийся, x_2),$$

$$\varphi_2 = \varphi_{21} \wedge \varphi_{22},$$

$$\varphi_1 \leftrightarrow \varphi_2.$$

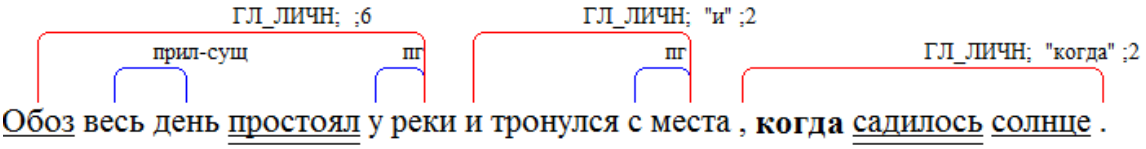
5. Система ДИАЛИНГ

Идеи, рассмотренные в предыдущих разделах, могут быть применены для анализа перефразированных вариантов предложений, а именно, оценивания их близости, также на основе диаграмм, получаемых на выходе системы ДИАЛИНГ. Система ДИАЛИНГ разрабатывалась как система русско-английского перевода с 1999 г. по 2002 г на базе ООО Диалинг (г. Москва) [2]. В разное время в работе над системой принимали участие более 20 специалистов, большинство из которых – известные ученые-лингвисты. Как и все современные системы обработки текста, ДИАЛИНГ включает в себя все основные этапы анализа текста.

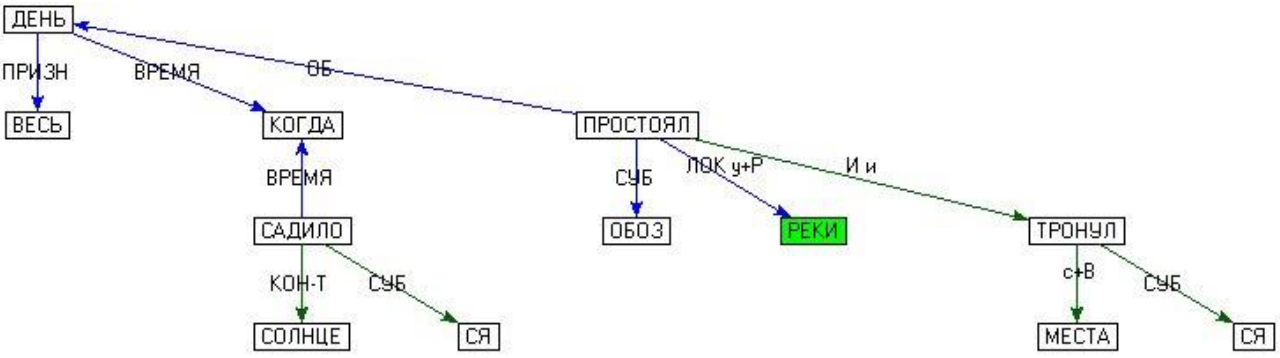
За основу системы автоматического русско-английского перевода ДИАЛИНГ была взята система французско-русского автоматического перевода (ФРАП), разработанная в ВЦП совместно с МГПИИЯ им. М. Тореза в 1976–1986 гг., и система анализа политических текстов на русском языке ПОЛИТЕКСТ, разработанная в центре информационных исследований в 1991–1997 гг. Ниже приведены примеры синтаксического и семантического анализа предложений, являющихся перефразированиями.

1. *Обоз весь день простоял у реки и тронулся с места, когда садилось солнце*.

1.1. Результат синтаксического анализа.



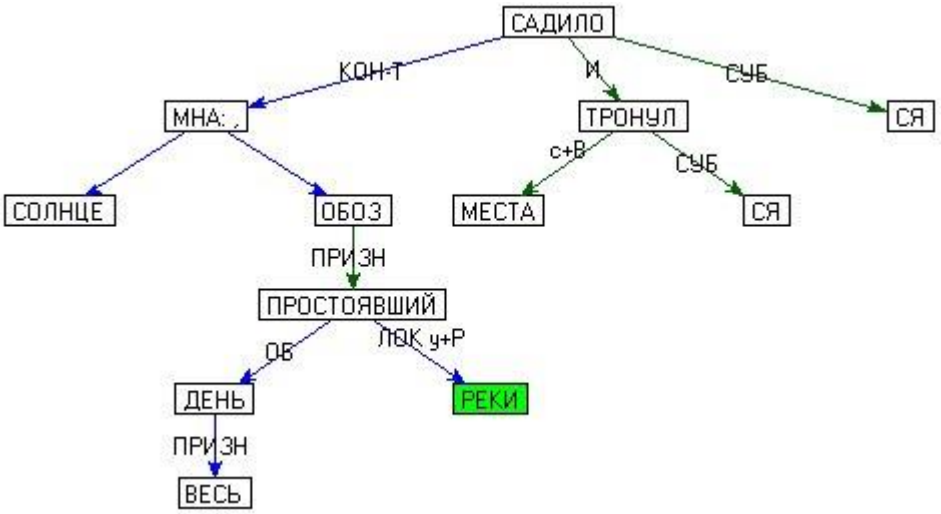
1.2. Результат семантического анализа.



2. *Садилось солнце, обоз, простоявший весь день у реки, тронулся с места*.

2.1. Результат синтаксического анализа.

2.2. Результат семантического анализа.



6. Заключение

В работе описан процесс создания базы знаний, содержащей информацию о временных и пространственных понятиях, встречающихся в текстах на естественном языке. Формально описан процесс обнаружения перефразированных предложений, содержащих некоторые пространственные и временные понятия. В настоящее время ведется работа по наполнению базы знаний. В результате проведенного исследования был сделан вывод о целесообразности применения таких инструментов, как Link Grammar Parser и ДИАЛИНГ, для решения поставленной задачи.

Link Grammar Parser – достаточно необычная система, главной причиной, по которой анализатор называют семантической системой, можно считать уникальный по полноте набор связей. Например, выделяются следующие классы наречий: ситуационные наречия, которые относятся ко всему предложению в целом (clausal adverb); наречия времени (time adverbs); вводные наречия, которые стоят в начале предложения и отделены запятой (openers); наречия, модифицирующие прилагательные, и т.д. Из достоинств системы необходимо отметить, что организация самой процедуры нахождения вариантов синтаксического представления очень эффективна.

Отметим также отрицательные моменты. Практическое тестирование системы показывает, что при анализе сложных предложений, длина которых превышает 25–30 слов, возможен комбинаторный взрыв, и результатом работы анализатора становится «панический» граф – как правило, случайный вариант синтаксической структуры, с лингвистической точки зрения неадекватной. Использование системы затруднено для флективных языков типа русского ввиду значительно возрастающего объема словарей, которые возникают в силу морфологической развитости флективных языков.

Для случая английского языка было проведено тестирование [4]. Основная задача, которая решалась – это определение релевантности текста поисковому запросу. В этом случае был обнаружен минимальный вариант, дававший достаточно хорошие результаты, когда учитывались всего 8 связей. Для английского языка был составлен перечень наиболее важных связей системы Link Grammar Parser. Он содержит 35 связей. Обнаружены связи, которые существенно портили ситуацию в случае неполных предложений, – всего 8 таких связей. Для русского языка такой детальный анализ связей не проведен. Возможности системы ДИАЛИНГ еще менее изучены.

Тем не менее, следует отметить, что диаграммы, полученные на выходе систем Link Grammar Parser и ДИАЛИНГ, представляют собой чрезвычайно интересный материал для

дальнейших исследований, в том числе для исследования временных и пространственных понятий как с точки зрения приложений, так и для теоретической лингвистики.

Литература

1. *Мельчук И. А.* Опыт теории лингвистических моделей типа «Смысл – Текст». М.: Наука, 1974. 315 с.
2. *Сокирко А. В.* Семантические словари в автоматической обработке текста // Канд. дисс., МГПИИЯ, Москва. 2000. 108 с.
3. Link Grammar Documentation, 2015, http://www.abisource.com/projects/link-grammar
4. *Батура Т. В., Бакиева А. М., Еримбетова А. С. И др.* Грамматика связей, релевантность и определение тем текстов: монография. Новосибирск: Изд-во СО РАН. 2018. 91 с.

Статья поступила в редакцию 27.08.2019;
переработанный вариант – 27.09.2019.

Батура Татьяна Викторовна
к.ф.-м.н., с.н.с., Институт систем информатики им. А. П. Ершова СО РАН (630090, Новосибирск, просп. Академика Лаврентьева 6), e-mail: tbatura@iis.nsk.su.

Ефимова Любовь Валерьевна
аспирант, Институт систем информатики им. А. П. Ершова СО РАН.

Еримбетова Айгерим Сембековна
аспирант, Новосибирский государственный университет, e-mail: aigerian@mail.ru.

Касекеева Айсулу Бесеновна
аспирант, Евразийский национальный университет им. Л. Н. Гумилева, Казахстан.

Мурзин Федор Александрович
к.ф.-м.н., зам. директора по научной работе, Институт систем информатики им. А. П. Ершова СО РАН, e-mail: murzin@iis.nsk.su.

Temporal and spatial concepts in natural language texts and their investigation

T. V. Batura, L. V. Efimova, A. S. Yerimbetova, A. B. Kasekeeva, F. A. Murzin

The aim of the work is to create a knowledge base containing information on temporal and spatial concepts found in natural language texts. The content of the base: the most important concepts related to time and space, from the S. I. Ozhegov explanatory dictionary; rephrased sentences; the results of the analysis of vocabulary articles (interpretations) and examples from a literature by means of Link Grammar Parser and DIALING software systems, etc. Results of the work can be used in intelligent information retrieval systems. The diagrams obtained at the output of Link Grammar Parser and DIALING systems are extremely interesting material for further research. It is advisable to investigate the possibility of using a number of constructions and concepts of mathematical logic in computer linguistics, such as: the construction of L. Henkin, realizability and omitting of types, model completeness, forcing, as well as a number of non-classical logics.

Keywords: computational linguistics, semantics, temporal and spatial concepts, Link Grammar Parser, DIALING.

УДК 577.218

Использование графических ускорителей для выявления функциональных сигналов в регуляторных районах дифференциально экспрессирующихся генов AGRP нейронов гипоталамуса мыши в ответ на голодание

А. В. Бочарников, Е. В. Игнатьева, О. В. Вишневский[1]

Выявление *de novo* контекстных сигналов в регуляторных районах генов эукариот существенно затрудняется как огромными объемами анализируемых выборок последовательностей, так и гигантским разнообразием контекстных сигналов. Нами предложен новый алгоритм оценки представленности вырожденных олигонуклеотидных мотивов, записанных в 15-буквенном IUPAC коде, в выборке нуклеотидных последовательностей и показана его высокая производительность по сравнению с ранее предложенным подходом. Данный метод основан, во-первых, на использовании деревьев префиксов, во-вторых, на соответствии префиксов мотивов диапазонам хешей в хешированных нуклеотидных последовательностях анализируемой выборки и, в-третьих, на технологии CUDA, позволяющей использовать для массового параллельного счета графические ускорители, широкодоступные для исследователей.

Предложенный подход был использован для проведения контекстного анализа промоторных областей генов мыши с достоверно изменившейся после лишения пищи экспрессией в AGRP (Agouti Related Peptide) нейронах гипоталамуса. Когда животное лишено пищи, так называемые AGRP нейроны гипоталамуса вырабатывают молекулы, которые повышают аппетит и облегчают набор веса. Понимание клеточных механизмов, лежащих в основе функционирования нейронов AGRP в ответ на потерю веса, необходимо для разработки методов борьбы с ожирением, которое является наследственным заболеванием, имеющим лишь несколько безопасных и долгосрочных эффективных методов лечения и стратегий вмешательства. Проведенный нами анализ выявил значимые олигонуклеотидные мотивы, ассоциированные с голоданием.

Ключевые слова: олигонуклеотидный мотив, GPGPU, ожирение.

1. Введение

Выявление функциональных сигналов в регуляторных районах генов является важной задачей современной биоинформатики и необходимо для понимания базовых механизмов регуляции транскрипции и трансляции.

Выявление олигонуклеотидных мотивов *de novo* является одним из наиболее ранних и широко используемых биоинформатических подходов к обнаружению контекстных сигналов в регуляторных районах генов. Такие мотивы могут соответствовать как сайтам связывания транскрипционных факторов, так и определенным физико-химическим особенностям регуляторных районов. В то же время их использование затруднено гигантским разнообразием возможных вариантов. Так, мотив длиной в 8 нуклеотидов, записанный в 15-буквенном IUPAC

[1] Благодарности: Работа поддержана бюджетным проектом № 0324-2019-0040.

коде (табл. 1), составляет $15^8 \sim 2.5 \times 10^9$ различных вариантов записи в 4-буквенном коде. Это заставляет исследователей использовать различные эвристические подходы, основанные на анализе частот *l*-плетов [1], деревьев суффиксов [2], локальном множественном выравнивании [3], EM методе (Expectation–Maximization) [4, 5] и т.д. В то же время такие подходы не гарантируют нахождения наиболее достоверного мотива. Решением может являться использование полнопереборного строкового подхода, что требует применения высокопроизводительных массивно-параллельных вычислительных систем, таких как GPU-ускорители [6].

Ранее нами был предложен такой полнопереборный подход, реализованный в виде программной системы Argo_CUDA [7] как на CPU, так и на GPU-устройствах и продемонстрирована его высокая эффективность в сравнении с существующими подходами. В данной работе мы предлагаем новый подход, основанный на использовании дерева префиксов мотивов и позволяющий существенно увеличить производительность.

Предложенный подход был использован для анализа промоторов генов, экспрессия которых достоверно отличалась в AGRP нейронах гипоталамуса голодающих и сытых мышей. В промоторах генов, повышающих экспрессию в ответ на голод и понижающих ее, были найдены достоверные олигонуклеотидные мотивы. Анализ полученных мотивов выявил достоверное сходство с рядом известных сайтов связывания транскрипционных факторов. Классификация генов, промоторы которых содержали сигналы, найденные с использованием системы DAVID [8, 9], выявила их достоверную ассоциацию с голоданием.

2. Материалы и методы

2.1. Описание алгоритма поиска мотивов, основанного на дереве префиксов мотивов

Для полнопереборной оценки представленности каждого из 15^k вырожденных мотивов длины k в выборке из N нуклеотидных последовательностей длины L необходимо сравнить каждый мотив длины k с каждой k-символьной подстрокой этой выборки, что потребует $N \times (L - k + 1)$ сравнений. Для ускорения этого процесса удобно записывать и нуклеотидные последовательности, и мотивы в виде бинарных хэшей (табл. 1).

Таблица 1. Бинарное представление вырожденных олигонуклеотидов записанных в 15-буквенном IUPAC коде

IUPAC код	Нукл.	Хэш
A	A	0001
T	T	0010
W	A/T	0011
G	G	0100
R	A/G	0101

IUPAC код	Нукл.	Хэш
K	T/G	0110
D	A/T/G	0111
C	C	1000
M	A/C	1001
Y	T/C	1010

IUPAC код	Нукл.	Хэш
H	A/T/C	1011
S	G/C	1100
V	A/G/C	1101
B	T/G/C	1110
N	A/T/G/C	1111

В этом случае для записи одного символа требуется 4 бита, а сравнение мотива с 8-нуклеотидной подстрокой возможно с использованием одной операции AND (табл. 2).

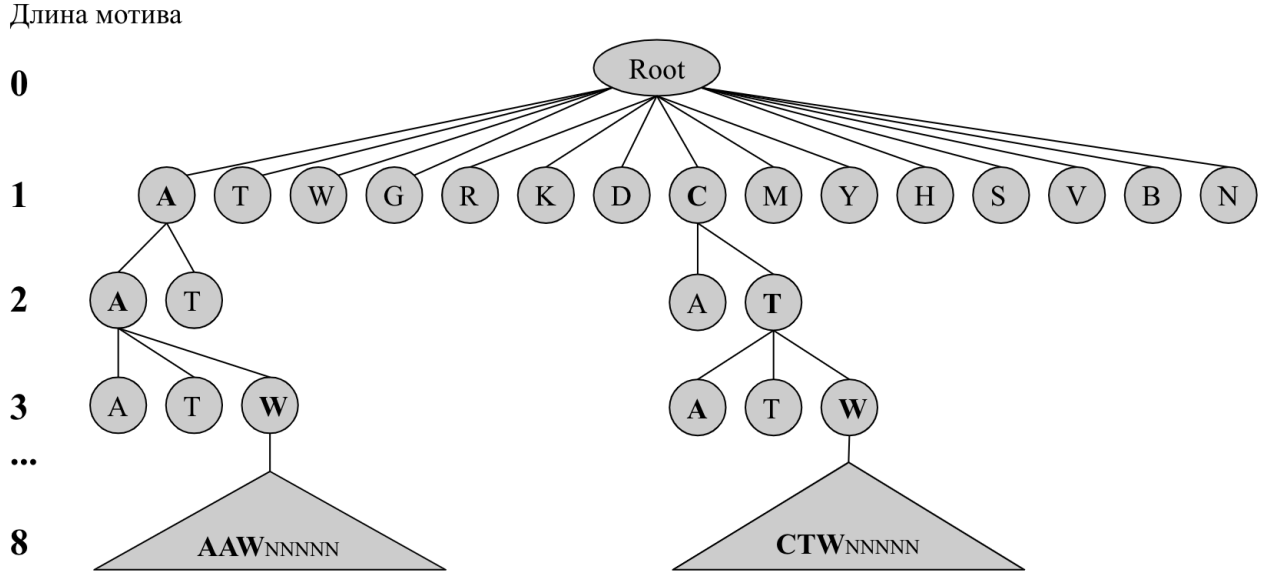
Таблица 2. Сравнение мотива WWWWAAAA и олигонуклеотида ATATAAAA

```
          boolean match = (hash & motif_hash) == hash

motif_hash   WWWWAAAA   0011 0011 0011 0011 0001 0001 0001 0001
hash         ATATAAAA   0001 0010 0001 0010 0001 0001 0001 0001
match        true       0001 0010 0001 0010 0001 0001 0001 0001
```

Если хотя бы в одной позиции мотив не совпадает с олигонуклеотидом, то результат побитового AND будет равен нулю. Таким образом, сложность сравнения мотива с подстрокой снижается с 8 операций посимвольного сравнения при стандартной записи до одной операции побитового AND при записи в виде хэша.

Полный перебор всех 15^k вариантов мотивов можно представить в виде обхода дерева (рис. 1). Корневому узлу соответствует пустая строка. У каждого узла, кроме листовых, имеется 15 потомков, в каждом из которых хранится символ 15-буквенного IUPAC алфавита. В каждом узле текущий префикс мотива определяется набором символов из всех узлов на пути от корня к текущему узлу. Для каждой вершины поддерево со всеми узлами-потомками содержит все возможные варианты мотивов с заданным префиксом. В листовых вершинах префиксом является полный мотив.



Поддерево мотивов с префиксом **AAT** Поддерево мотивов с префиксом **CTW**

Рис. 1. Представление множества мотивов в виде дерева степени 15 глубины 8

Мотив совпадает с подстрокой входной выборки только в том случае, если их префиксы совпадают. Таким образом, каждому возможному префиксу мотива соответствует множество подстрок входной выборки нуклеотидных последовательностей (рис. 2).

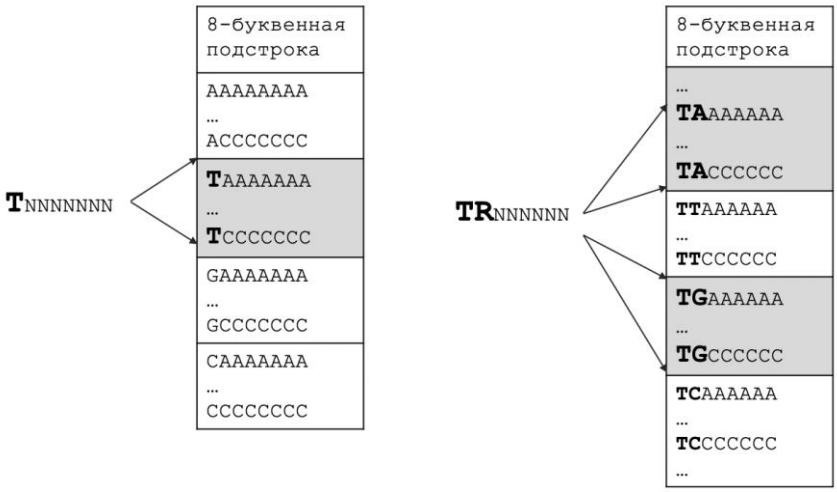
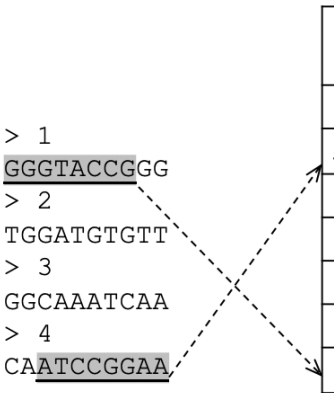


Рис. 2. Пример сопоставления вырожденных мотивов (с префиксами T и TR) и олигонуклеотидных диапазонов во входной выборке последовательностей с сохранением номера последовательности, в которой эти мотивы предоставлены

Для сокращения числа сравнений каждого мотива со всеми подстроками входной выборки был предложен метод построения диапазонов подстрок по префиксу мотива длины l_p, состоящий из трех шагов.

Шаг 1. Представление входной выборки в виде набора отсортированных хэшей.

Входные последовательности представляются в виде множества всех входящих в них k-буквенных подстрок. После чего подстроки преобразуются в единый список хэшей и сортируются в лексикографическом порядке (рис 3).



8-буквенная подстрока	Номер последовательности	Бинарное представление (хэш)
AATCCGGA	4	0001 0001 0010 1000 1000 0100 0100 0001
ATCCGGAA	4	0001 0010 1000 1000 0100 0100 0001 0001
TGGATGTG	2	0010 0100 0100 0001 0010 0100 0010 0100
GATGTGTT	2	0100 0001 0010 0100 0010 0100 0010 0010
GTACCGGG	1	0100 0010 0001 1000 1000 0100 0100 0100
GGATGTGT	2	0100 0100 0001 0010 0100 0010 0100 0010
GGTACCGG	1	0100 0100 0010 0001 1000 1000 0100 0100
GGGTACCG	1	0100 0100 0100 0010 0001 1000 1000 0100
GGCAAATC	3	0100 0100 1000 0001 0001 0001 0010 1000
GCAAATCA	3	0100 1000 0001 0001 0001 0010 1000 0001
CAAATCAA	3	1000 0001 0001 0001 0010 1000 0001 0001
CAATCCGG	4	1000 0001 0001 0010 1000 1000 0100 0100

Рис. 3. Пример построения отсортированного в лексикографическом порядке списка хэшей всех 8-буквенных подстрок по входной выборке нуклеотидных последовательностей длины 10

Шаг 2. Построение диапазонов для нуклеотидных префиксов (в 4-букв. коде).

На втором шаге перебираются все возможные олигонуклеотидные префиксы заданной длины (строка 3). Для каждого префикса с помощью бинарного поиска определяется диапазон во входной выборке, где он представлен (строки 5–6).

```
1 total_prefixes = pow(4, prefix_size)
2 Ranges ranges_by_olig_prefix
3 for (i = 0; i < total_prefixes; i++)
4    olig_prefix_hash = olig_idx_to_hash(i)
5    low_bound= lower_bound(sequences, olig_prefix_hash)
6    up_bound = upper_bound(sequences, olig_prefix_hash | 0xFFFFFFFF)
7    if (low_bound != up_bound)
8        ranges_by_olig_prefix[olig_prefix_hash].add({ low_bound, up_bound })
```

Здесь *prefix_size* – длина префикса мотива l_p, *olig_idx_to_hash* – функция, которая по индексу олигонуклеотида возвращает его представление в виде хэша (табл. 2), *lower_bound* – позиция первого элемента, который не меньше заданного, *upper_bound* – позиция первого элемента, который больше заданного, *sequences* – отсортированный список хэшей всех подстрок длины *k* входной выборки.

Шаг 3. Построение диапазонов для префиксов мотивов (в 15-букв. IUPAC коде).

На третьем шаге перебираются все возможные варианты префиксов мотивов заданной длины (строка 3). Для каждого из префиксов мотива рассматриваются все варианты его записи в виде олигонуклеотида (строка 5) и для них объединяются диапазоны, посчитанные в шаге 2 (строка 8).

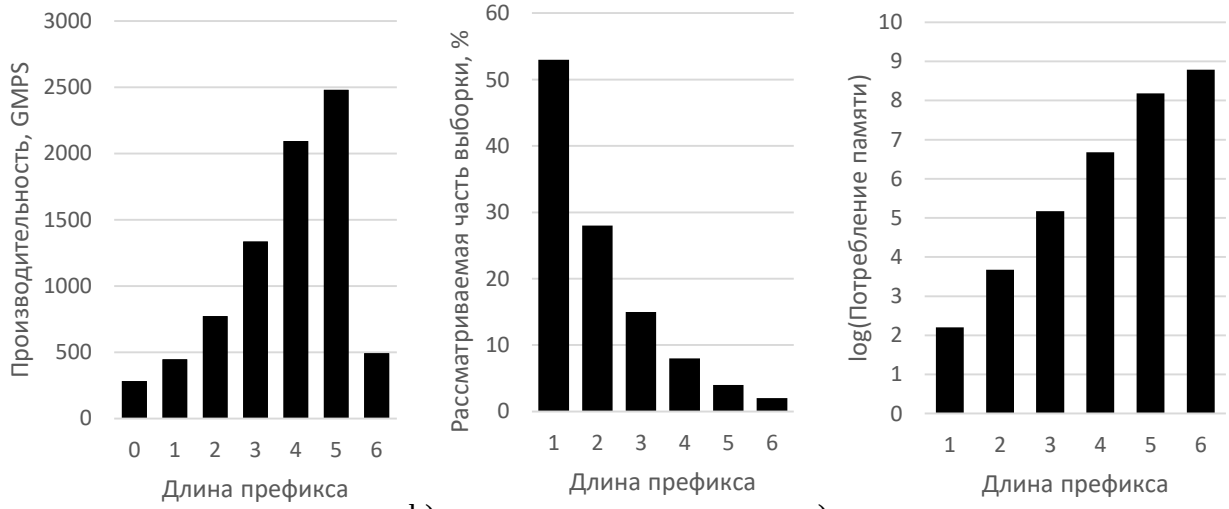
```
1 total_prefixes = pow(16, prefix_size)
2 Ranges ranges_by_motif_prefix
3 for (i = 1; i < total_prefixes; i++)
4     motif_prefix_hash = motif_idx_to_hash(i, prefix_size)
5     for (j = 0; j < motif_variants(i); j++)
6         olig_hash_mask = motif_variant(motif_prefix_hash)
7         range = ranges_by_olig_prefix[olig_hash_mask]
8         ranges_by_motif_prefix[motif_prefix_hash].add(range)
```

Здесь *motif_idx_to_hash* – функция, которая по индексу мотива возвращает его представление в виде хэша.

После того, как получены диапазоны для всех возможных префиксов мотивов, они могут быть использованы для оценки представленности мотивов в выборке:

```
1 for (i = 0; i < pow(15,8); i++)
2     motif_hash = motif_idx_to_hash(i)
3     motif_prefix = motif_hash >> (32 - 4*prefix_size)
4     motif_ranges = ranges.get(motif_prefix)
5     for (j = 0; j < ranges.size; j++)
6         for (k = motif_ranges[j].start; k < motif_ranges[j].end; k++)
7             match = (sequence_hashes[k] & motif_hash) == sequence_hashes[k])
```

Таким образом, число сравнений мотива с подстроками входной выборки существенно снижается. Проведенный нами анализ показал, что количество перебираемых подстрок уменьшается (рис. 4б), а производительность (рис. 4а) и размер оперативной памяти (рис. 4с) для хранения диапазонов растут в зависимости от длины префикса. Оптимальным размером префикса является 5 или 6 в зависимости от размеров входной выборки.



a) b) c)

Рис. 4. Оценка (a) производительности GMPS (Giga matches per second, ось Y), (b) рассматриваемый процент подпоследовательностей в выборке и (c) размер оперативной памяти для хранения диапазонов в зависимости от длины префикса (ось X) для выборки из 1000 последовательностей длины 128

2.2. Реализация алгоритма поиска мотивов на GPU

Современные GPU-ускорители содержат тысячи вычислительных ядер, позволяющих одновременно обсчитывать десятки тысяч потоков. При этом наиболее эффективно с помощью GPU могут быть распараллелены задачи, где отсутствуют зависимости по данным. К таким задачам может быть отнесен и расчет представленности мотива в выборке нуклеотидных последовательностей.

```
1  motif_index = blockIdx.x * blockDim.x + threadIdx.x
2  motif_hash = idx_to_hash(motif_index)
3  extern __shared__ unsigned char matched_sequences[]
4  motif_prefix = motif_hash >> (8 - prefix_size)*4
5  Range range = ranges[motif_prefix]
6  offset = threadIdx.x * sequences_count

8  for (i = range.start_pos; i < range.end_pos; i++)
9      hash = sequence_hashes[i]
10     match = (hash & motif_hash) == hash
11     if (match)
12         seq_id = hash_to_sequence_id[i]
13         matched_sequences[offset + seq_id] = 1
14
15 for (i = offset; i < (offset + sequences_count); i++)
16     if (matched_sequences[i]) presense += 1
17         motif_presense[motif_index] = presense
```

При расчетах на GPU каждый поток в блоке формирует очередной индекс мотива (строка 2) и с помощью функции *idx_to_hash* преобразует его в хэш. Префикс вычисляется с помощью битового сдвига хэша мотива на *(8-prefix_size)*4* бит вправо (строка 4). Полученный префикс используется для получения соответствующего ему диапазона подстрок входной выборки (строка 5). По индексу каждой подстроки, совпадающей с мотивом (строка 10), в массив результатов *matched_sequence* записывается единица (строка 13). Каждому потоку из блока выделяется свой участок памяти для результатов размером *sequence_count* (строки 6, 13, 15). Встречаемость мотива во входной выборке оценивается количеством единиц в массиве результатов (строки 16–17). Для ускорения сохранения результатов сравнений массив *matched_sequences* расположен в разделяемой памяти GPU (строка 3). Размер S разделяемой памяти ограничен (48–96 Кб на блок), поэтому максимальное число последовательностей, обрабатываемых за 1 запуск вычислительного ядра, составляет $S \,/\, THREAD\_PER\_BLOCK$

3. Результаты и обсуждение

3.1. Реализация алгоритма поиска мотивов на GPU

Для того чтобы получить оценки производительности работы программы, не зависящие ни от длины и количества анализируемых последовательностей, ни от длины мотивов, будем измерять производительность в количестве сравнений позиций мотивов с позициями выборки последовательностей за единицу времени. Для набора мотивов M и выборки последовательностей D производительность GMPS вычисляется следующим образом:

$$\boldsymbol{GMPS = \frac{|M| \times |D|}{t \times 10^9} = \frac{k \times N_{mot} \times N_{seq} \times (L - k + 1)}{t \times 10^9}},$$

где $|M|$ – суммарное количество всех букв в наборе вырожденных олигонуклеотидных мотивов, $|D|$ – суммарное количество всех позиций в выборке последовательностей, с которыми возможно сравнение с мотивами, t – время работы в секундах.

С использованием предложенной меры мы провели оценку эффективности использования разработанного метода на двух графических ускорителях: NVidia GTX 1070 и NVidia Tesla v100. На рис. 5 приведены сравнительные оценки производительности на различных вычислительных платформах на выборке в зависимости a) от длины последовательностей и b) от их количества. Длина префикса – 5.

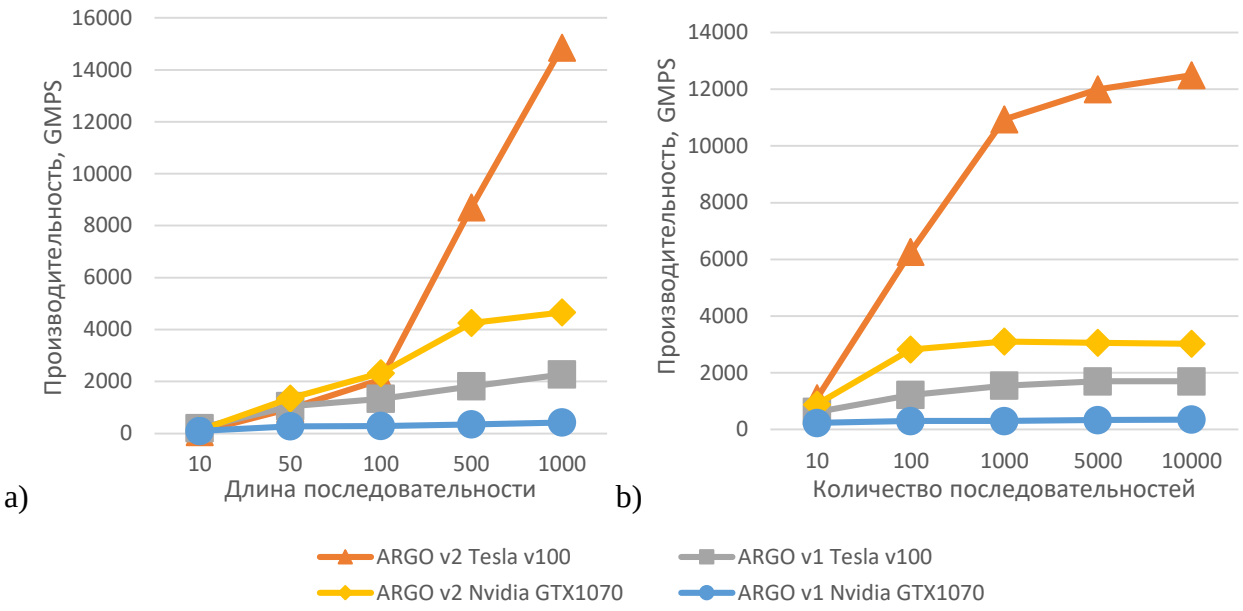


Рис. 5. Оценка производительности GMPS (Giga matches per second, ось Y)
на различных устройствах в зависимости
a) от длины последовательностей (ось X), количество последовательностей – 100;
b) от количества последовательностей (ось X), длина последовательностей – 128

Из рис. 5a и 5b хорошо видно, что NVidia Tesla v100, обладающая 5120 процессорами и более современной архитектурой, имеет в среднем в 4 раза более высокую производительность по сравнению с NVidia GTX 1070 (2048 ядер). Отметим, что оба графические ускорителя имеют худшую производительность при малых длинах анализируемых последовательностей и их малом количестве. Это можно объяснить тем, что внутренний цикл становится достаточно коротким и запуск вычислительного ядра происходит чаще. Можно отметить, что при количестве последовательностей от 1000 производительность обоих GPU стабилизируется. При этом производительность системы растет с увеличением размера выборки.

Рис. 5 ясно демонстрирует, что на всех рассмотренных GPU-устройствах при всех используемых параметрах анализируемых выборок предложенный нами новый алгоритм оценки представленности мотивов обеспечивает в среднем десятикратный прирост производительности.

3.2. Выявление олигонуклеотидных мотивов в промоторах генов голодающих мышей

Список дифференциально экспрессирующихся генов мышей был получен из [10]. Он содержал 968 генов, повышающих экспрессию в ответ на голодание, и 610 генов, понижающих экспрессию. Из базы данных Ensembl [11] с использованием системы BioMart [12] были получены последовательности промоторов в районе [-200; +1] относительно старта транскрипции генов, повышающих (выборка Neg) экспрессию в ответ на голодание и понижающих экспрессию (выборка Pos). Обе выборки были проанализированы с использованием предложенного нами нового метода.

Анализ полученных олигонуклеотидных мотивов с известными регуляторными мотивами проводился с помощью системы Tomtom [13]. Функциональная аннотация генов, промоторы которых содержали найденные сигналы, проводилась с использованием системы DAVID [8, 9].

Проведенный анализ показал, что мотивы выборки Neg имеют достоверное сходство с такими сайтами связывания транскрипционных факторов (ССТФ), как NFYA, FOXI1, CEBPZ, NFYB, NFYC, PBX3, SP4, ZN335, SP2, а мотивы выборки Pos имеют достоверное сходство с такими ССТФ, как SP1, E2F4, EGR1, SALL4, GLI2, KLF8, SRBP2, KLF5, ARNT2, ZIC1, HTF4, E2F7, PLAG1.

Функциональная аннотация генов, промоторы которых содержали мотивы, найденные в выборке Neg, показала их достоверную ассоциацию с GO терминами ответа на эндоплазматический ретикулярный стресс (ERS). В их число входили такие хорошо известные гены – регуляторы или эффекторы ответа на ERS, как синоволин 1 (Syvn1), hypoxia up-regulated 1 (Hyou1) и белки теплового шока (Hspa5, Hsp90b1).

Мотивы из выборки Pos, найденные в генах, достоверно ассоциированы с широким кругом биологических процессов, связанных с нормальным развитием и функционированием нейронов.

4. Заключение

Нами предложен новый высокопроизводительный полнопереборный алгоритм для выявления вырожденных олигонуклеотидных мотивов в больших выборках нуклеотидных последовательностей, основанный на дереве префиксов мотивов. С его помощью в промоторах дифференциально экспрессирующихся генов мышей были выявлены контекстные сигналы, достоверно ассоциированные с голоданием.

Литература

1. *Pesole G, Liuni S, Dsouza M.* PatSearch: A pattern matcher sontware that finds functional elements in nucleotide and protein sequences and assesses their statistical significance // Bioinformatics. 2000. V. 16, № 5. P. 439–450.
2. *Marsan L, Sagot M. F.* Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification // J Comput Biol. 2000. V. 7 (3–4). P. 345–362.
3. *Hertz G, Stormo G.* Identifying DNA and protein patterns with statistically significant alignments of multiple sequences // Bioinformatics. 1999. V. 15 (7–8). P. 563–577.
4. *Grundy W. N., Bailey T. L., Elkan C. P.* ParaMEME: A parallel implementation and a web interface for a DNA and protein motif discovery tool // CABIOS. 1996. V. 12. P. 303–310.
5. *Lawrence C. E., Altschul S. F., Boguski M. S., Liu J. S. et al.* Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment // Science. 1993. V. 262, № 5131. P. 208–214.
6. *Nickolls J., Buck I., Garland M., Skadron K.* Scalable Parallel Programming with CUDA // Queue. 2008. V. 6, № 2. P.40–53.
7. *Vishnevsky O. V., Bocharnikov A. V., Kolchanov N. A.* ARGO_CUDA: Exhaustive GPU based approach for motif discovery in large DNA datasets // Journal of Bioinformatics and Computation Biology. 2017. V. 16, № 1.
8. *Huang D. W., Sherman B. T., Lempicki R. A.* Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources // Nature Protoc. 2009. V. 4, № 1. P. 44-57.
9. *Huang D. W., Sherman B. T., Lempicki R. A.* Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists // Nucleic Acids Res. 2009. V. 37, № 1. P. 1–13.
10. *Henry F. et al.* Cell type-specific transcriptomics of hypothalamic energy-sensing neuron responses to weight-loss // Elife. 2015. Sep 2–4.
11. *Zerbino D. R., Flicek P. et al.* Ensembl 2018 // PubMed. 2018. PMID: 29155950.
12. *Durinck S., Spellman P., Birney E., Huber W.* Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt // Nature Protocols. 2009. V. 4. P. 1184–1191.
13. *Gupta S., Stamatoyannopoulos J. A., Bailey T. L., and Noble W. S.* Quantifying similarity between motifs // Genome Biology. 2007. V. 8, № 2. R24.

Статья поступила в редакцию 26.07.2019;
переработанный вариант – 30.09.2019.

Бочарников Андрей Васильевич
аспирант, Институт систем информатики им. А. П. Ершова СО РАН (630090, Новосибирск, пр. Академика Лаврентьева, 6), e-mail: andrey.bocharnikov@gmail.com.

Игнатьева Елена Васильевна
к.б.н, старший научный сотрудник, Институт цитологии и генетики СО РАН (630090, Новосибирск, пр. Академика Лаврентьева, 10); доцент, Новосибирский государственный университет, e-mail: eignat@bionet.nsc.ru.

Вишневский Олег Владимирович
к.б.н, научный сотрудник, Институт цитологии и генетики СО РАН; ст. преподаватель, Новосибирский государственный университет, e-mail: oleg@bionet.nsc.ru

GPU-based algorithm for context analysis of the core promoter region of mouse genes differently expressed in hypothalamic energy-sensing neurons in response to weight-loss

A. Bocharnikov, E. Ignatieva, O. Vishenvskiy

De novo motif discovery in the regulatory regions of eukaryotic genes poses a complex computational problem due to the large size of datasets and huge diversity of motifs. This article suggests a new algorithm for measuring the presence of degenerate oligonucleotide motifs written as a 15-letter IUPAC code in a DNA dataset. Its performance has increased 10 times compared with the previous one. There are three key ingredients of this method. The first one is the prefix trees. The second is the relation between motif prefixes and hash ranges in the analyzed nucleotide sequences. The third consists of applying CUDA framework to the massive parallelization allowing to use affordable graphic accelerators.
The context analysis of promoter regions of mouse genes differently expressed (DEG) in hypothalamic AGRP neurons after food deprivation was performed with the proposed method. When an animal is deprived of food, AGRP neurons produce molecules that increase appetite and stimulate weight gain. The understanding of how AGRP neurons respond to weight loss is important to confront the obesity. Nowadays, this hereditary disease lacks methods of treatment and intervention strategies which would be both safe and efficient in the long term. The performed analysis revealed relevant oligonucleotide motifs that were associated with starvation.

Keywords: oligonucleotide motif, GPGPU, obesity.

УДК 004.94

Автоматизация исследований развития опорной транспортной сети

М. А. Бульонков, Т. В. Нестеренко

Рассматривается система автоматизации научных исследований для решения задачи прогнозирования развития опорной транспортной сети России. Формальная модель транспортной сети допускает различные виды транспорта, продукты, узлы производства и потребления и сводится к задаче минимизации суммарной стоимости перевозок при совокупности набора линейных ограничений. Система предоставляет пользовательский интерфейс для задания и редактирования всех параметров транспортной сети. Особое внимание уделяется визуальному и интерактивному представлению результатов моделирования. Целостное восприятие результатов достигается за счёт отображения как входных данных (вид транспорта, пропускная способность, стоимость перевозки), так и результатов моделирования (перевозимый объём и загруженность плеча) непосредственно либо на карте, либо на схеме. При этом возможно отображение как отдельного продукта, так и всех вместе. На практике зачастую требуется решить задачу, обратную моделированию, например, выяснить, при каких тарифах объёмы перевозки по определённому плечу будут превосходить заданное значение. Для этого предлагается использовать стохастические методы, основанные на массовом решении транспортной задачи для множества варьируемых параметров, таких как пропускная способность транспортного плеча, тариф на перевозку груза или его обработку в транспортном узле. Методы кластеризации позволяют выделить из всего множества просчитанных вариантов относительно небольшое количество «типичных» решений. Это даёт возможность эксперту оценить как условия, так и вероятность прогнозируемой транспортной ситуации. Показано также, что такой подход позволяет автоматически выделять складывающиеся транспортные коридоры и определять зависимость объёмов перевозки по данному плечу от конкретного варьируемого параметра.

Ключевые слова: транспортная задача, моделирование, система автоматизации научных исследований.

1. Введение

В 2009 году творческий коллектив, состоящий из сотрудников СИМОиР, ИСИ СО РАН, ИЭиОПП СО РАН и НГУ, начал инициативный проект МИКС (Модельная Информационная Картографическая Система) [1], предусматривающий создание информационной среды, направленной на поддержку задач управления регионами и их взаимодействия между собой. В работе [2] была представлена система МИКС-ПРОСТОР, основным назначением которой являлась автоматизация экономических экспериментов, связанных с прогнозированием развития опорной транспортной сети. Эта система позволяет задать прогноз изменения параметров, описывающих услуги по перевозке грузов и их обработке в транспортных узлах, и решить задачу оптимальной перевозки в условиях конкуренции нескольких видов транспорта. Таким образом, мы получаем вариант развития опорной транспортной сети, обеспечивающий рациональное взаимодействие входящих в нее видов транспорта. Иными словами, мы решаем задачу прогнозирования путём имитационного моделирования типа «что-если» [3].

2. Моделирование типа «что-если»

На первом этапе задача состояла в основном в автоматизации процесса научных исследований. В тот момент у исследователей уже имелся весь необходимый аппарат для проведения экспериментов. Транспортная задача формулировалась как задача линейного программирования [4], в которой величинами являются объёмы и направление перевозки продуктов по плечам транспортной сети, коэффициентами – стоимость обработки грузов и длины плеч, а ограничения формулируются как неравенства вида:

- суммарный объём перевозимых по плечу продуктов вне зависимости от направления не должен превышать пропускной способности;
- объём ввозимого продукта должен быть равен сумме вывозимого и разгружаемого;
- объём потребляемого продукта должен быть не меньше объёма выгружаемого;
- и т.п.

Задача состоит в минимизации суммарных затрат на обработку грузов.

Рассмотрим формализованную постановку задачи.

Для задач прогнозирования развития транспортной сети используется экономическая модель [5], оперирующая следующими множествами:

P – множество всех видов груза,

T – множество всех видов транспорта,

R – множество всех транспортных узлов.

Каждое транспортное плечо определяется видом транспорта и парой узлов: (t, r, r'). Далее приведены ограничения, которые накладываются на параметры транспортной сети.

- Ограничения на погрузку и выгрузку:

$$\sum_t X_{tr}^p \le A_r^p, \quad \sum_t Y_{tr}^p \ge B_r^p,$$

где A_r^p – объем продукта p, произведенного в узле r;

X_{tr}^p – объем продукта p, погруженного на транспорт вида t в узле r;

B_r^p – объем продукта p, потребленного в узле r;

Y_{tr}^p – объем продукта p, выгруженного с транспорта вида t в узле r.

- Ограничения, связанные с обработкой в узле:

$$\sum_{r'} W_{tr'r}^p = Y_{tr}^p + \sum_{t'} V_{tt'r}^p + Z_{tr}^p, \quad X_{tr}^p + \sum_{t'} V_{tt'r}^p + Z_{tr}^p = \sum_{r'} W_{trr'}^p,$$

где Z_{tr}^p – объем продукта p, провезенного транзитом на t через узел r;

$V_{tt'r}^p$ – объем продукта p, перегруженного с транспорта t на транспорт t' в узле r;

$W_{tr'r}^p$ – объем продукта p, привезенного в узел r по плечу (t, r', r);

$W_{trr'}^p$ – объем продукта p, вывезенного из узла r по плечу (t, r, r').

- Ограничения, связанные с пропускными способностями $P_{trr'}$ по плечу (t, r, r'), которые учитывают перевозку по плечу в обоих направлениях:

$$P_{trr'} = P_{tr'r}\ , \ \sum_p (W_{trr'}^p + W_{tr'r}^p) \le P_{trr'}.$$

Целевая функция модели является суммой издержек, связанных с переработкой грузов в транспортных узлах и с перевозкой грузов по плечам:

$$\sum_{t,r,p} \vec{c}_t^p\, X_{tr}^p + \sum_{t,r,p} \overleftarrow{c}_t^p\, Y_{tr}^p + \sum_{t,r,p} \bar{c}_t^p\, Z_{tr}^p + \sum_{t,t',r,p} c_{tt'r}^p V_{tt'r}^p + \sum_{t,r,r',p} \varphi_{trr'}^p l_{trr'} W_{trr'}^p \to min,$$

где коэффициенты $\vec{c}_t^p$, $\overleftarrow{c}_t^p$, $\bar{c}_t^p$, $c_{tt'r}^p$ являются удельными (в пересчете на единицу груза) затратами на погрузку, выгрузку, обслуживание транзита и перегрузку с одного вида транспорта на другой соответственно, $l_{trr'}$ – длина плеча (t, r, r'), $\varphi_{trr'}^p$ – стоимость перевозки вдоль плеча единицы объема продукта на единицу расстояния.

Таким образом, мы имеем упрощенную постановку задачи, где коэффициенты при переменных $X_{tr}^p, Y_{tr}^p, Z_{tr}^p, V_{tt'r}^p, W_{trr'}^p$ являются константами.

Изначально исходные данные задавались в виде текстовых файлов стандартизованного формата и поступали на вход решателю, который представлял собой совокупность консольных программ [6]. Результат выдавался также в виде текстового файла. Таким образом, проведение единичного эксперимента могло занимать часы и даже дни.

Первоочередной задачей стало создание такого пользовательского интерфейса, который бы обеспечивал следующее:

- формирование транспортной сети: задание перечней видов транспорта и продуктов, узлов и плеч (дорог), их пропускной способности и длины;
- задание количества производимых и потребляемых в узлах продуктов, а также базовых тарифов перевозки продуктов по плечам и обработки в узлах (например, транзита или перегрузки с одного вида транспорта на другой);
- запуск решателя [7];
- визуализацию полученного результата.

Если для подготовки входных данных естественно использовать стандартные диалоговые возможности, то отображение результатов потребовало разработки специальных средств. Наиболее естественным представляется отображение транспортной сети на карте. По ряду соображений, обсуждение которых выходит за рамки данной работы, мы решили не привязываться к известным ГИС, даже если они и предоставляли программные интерфейсы для встраивания новой функциональности. Для наших целей и качественных требований оказалось достаточным решить проблему координатной привязки к «карте-подложке». Таким образом, система предоставляет на выбор одну из предобработанных карт, поверх которой отображаются узлы, плечи и прочие элементы транспортной сети.

На практике оказалось, что пространственное расположение на карте далеко не всегда удобно, например, в случае, когда «концентрация» узлов сильно отличается для разных регионов. Кроме того, дополнительная информация легче воспринимается, если плечи ортогональны и без изломов. Поэтому в дополнение к картам было разработано схематическое отображение сети, при которым приблизительно сохраняется взаимное расположение узлов. Грубо говоря, достаточно чтобы, Северный морской путь находился сверху, а Москва – левее Якутска, но необязательно, чтобы Диксон оказался правее Новосибирска. Два представления одной и той же транспортной сети показаны на рис. 1.

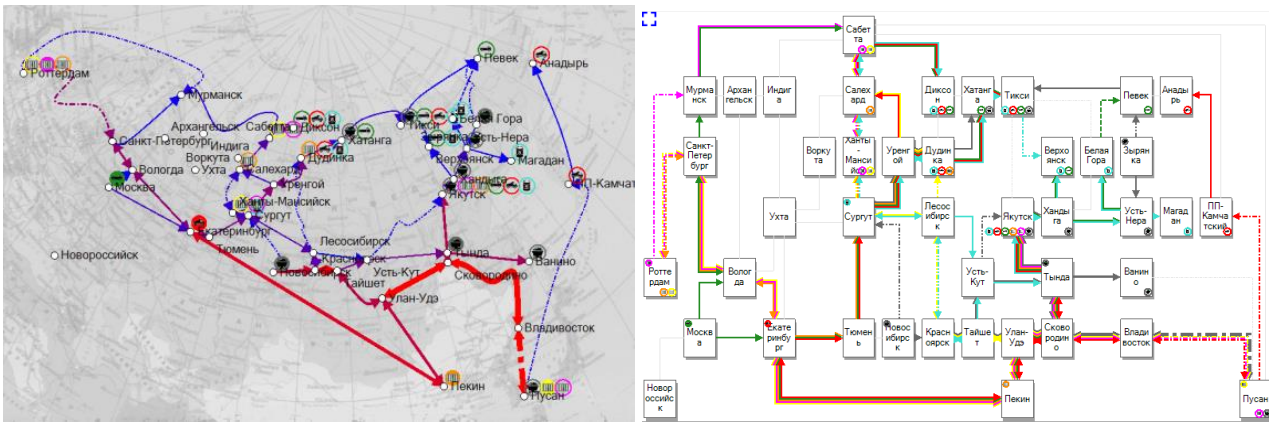


Рис. 1. Пространственное отображение транспортной сети

Для каждого плеча на карте или схеме требуется отобразить большое число параметров, таких как вид транспорта, пропускная способность, перевозимый объём и стоимость перевозки (в единицах на километр, для каждого вида продукта), загруженность плеча и т.д. При этом количество визуальных средств весьма ограничено: тип линии (сплошная, пунктирная и т.п.), её цвет и толщина. Использование всплывающих окон здесь не помогает, поскольку они не дают возможности получить общее представление о решении задачи. В текущей версии системы пользователь может выбрать один параметр, например, объём, который будет определять толщину и цвет линии, либо для всех продуктов вместе, либо для одного выбранного продукта.

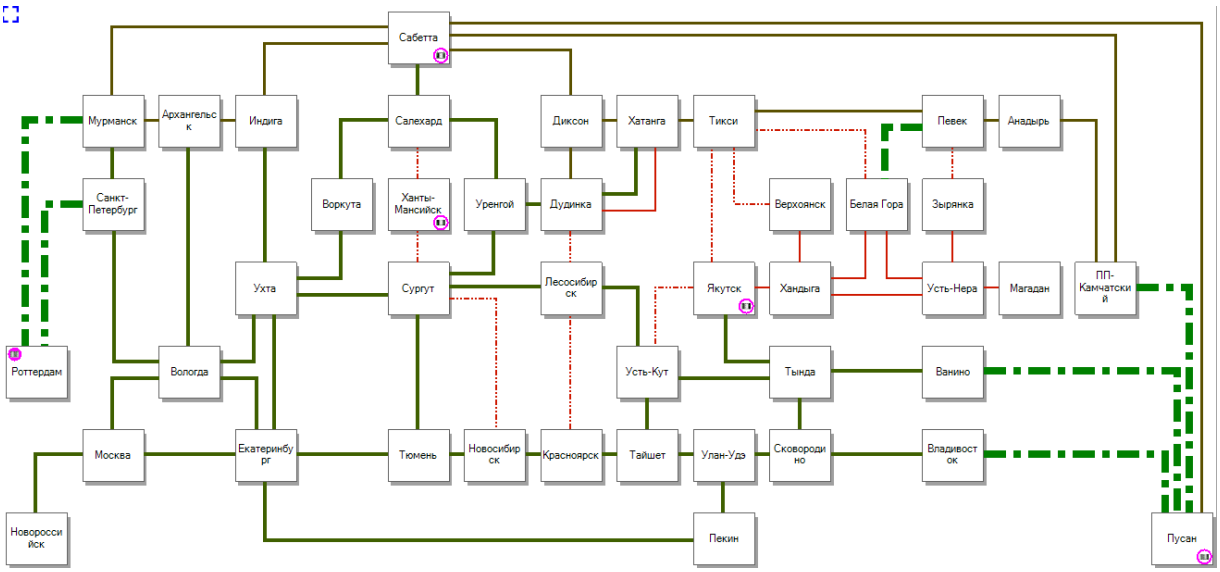


Рис. 2. Пространственное отображение пропускной способности плеч

Пример подобного отображения приведён на рис. 2, где в качестве отображаемого параметра плеча выбрана его пропускная способность. Зелёный цвет означает большую пропускную способность, серый – нейтральную, а красный – малую.

Естественно, пространственное отображение не является единственно возможным. Иногда полезным оказывается обычное табличное представление, особенно, если его дополнить упорядочением строк, соответствующих транспортным коридорам, типичными примерами которых являются Северный морской путь (СМП) или Транссибирская магистраль. Следует отметить, что понятие транспортного коридора может возникать не только априорно. Оно может появиться в зависимости от конкретного способа перевозки, выбранного в данном решении транспортной задачи для данного продукта. Причем в этот момент может изменяться и ориентация плеч: по одному и тому же плечу уголь везут в одну сторону, а контейнеры – в другую. Кроме того, клетки таблицы могут быть раскрашены теми же цветами, что и дороги на карте.

Плечо	Вид трансп	Средний объём	Объём	Сред. цена за 1км.	Цена за 1км.	Сред. загруже	Загруже	Зависим от Новосиб - Красноя
Дудинка-Лесосибир...	Река	2.22	2.00	1.44	1.30	0.22	0.20	0.58
Сургут-Новосибирск	Река	6.72	2.00	3.44	1.20	0.67	0.20	-0.55
Хандыга-Верхоянск	Авто	2.00	2.00	5.00	5.00	0.20	0.20	
Якутск-Усть-Кут	Река	2.00	2.00	3.00	3.00	0.20	0.20	
Мурманск-Роттердам	Море	3.22	3.00	0.64	0.60	0.03	0.03	0.58
ПП-Камчатский-Пус...	Море	3.00	3.00	0.60	0.60	0.03	0.03	
Усть-Нера-Белая Го...	Авто	4.00	4.00	9.00	9.00	0.40	0.40	
Мурманск-Сабетта	Лед	4.22	4.00	1.27	1.20	0.11	0.10	0.58
Хатанга-Тикси	Лед	5.00	5.00	13.00	13.00	0.13	0.13	
Ханты-Мансийск-Са...	Река	5.00	5.00	2.48	2.48	0.50	0.50	
Сургут-Ханты-Мансий...	Река	6.00	6.00	3.00	3.00	0.60	0.60	
Дудинка-Хатанга	Авто	6.00	6.00	16.00	16.00	0.60	0.60	
Хандыга-Усть-Нера	Авто	6.00	6.00	10.50	10.50	0.60	0.60	
Диксон-Сабетта	Лед	6.00	6.00	17.00	17.00	0.15	0.15	

Рис. 3. Табличное представление решения

Следующий вопрос, который возникает в задаче типа «что-если», – сравнение двух решений. Предположим, что определён некоторый базовый вариант транспортной сети, для которого было получено решение, и теперь исследователь хочет выяснить, что будет, например, если повысить в два раза пропускную способность по Транссибу или снизить на 15 % тариф

перевозки по СМП. Для этого необходимо просчитать данную ситуацию и сравнить её с базовым решением. Мы опробовали несколько способов сравнения двух решений: от простого табличного представления до пространственно-анимационного, при котором отображение на карте одного решения плавно сменяется на другое, и обратно. Это позволяет «мгновенно» оценить места, в которых решения различаются, и определить степень отличия.

3. Моделирование типа «как-чтобы»

Зачастую, с точки зрения эксперта, ответ на вопрос «что-если» носит вспомогательный характер. На самом деле его интересует некоторая цель и значения параметров транспортной сети, при которых эта цель достигается. При этом часто как сама цель, так и, возможно, изменяемые параметры локальны. Примерами вопросов, соответствующих таким целям, являются следующие:

- При каких тарифах данное плечо будет загружено по крайней мере на 25 %?
- Насколько имеет смысл увеличивать пропускную способность данного транспортного коридора?

Подобного рода задачи можно исследовать путём проведения множества экспериментов, отслеживая, как изменяется решение в зависимости от варьируемых параметров. Опять же, процесс этот трудоёмкий и ненадёжный, поскольку зависимость объёмов перевозки от тарифов и пропускной способности при их одновременном изменении не является линейной, она даже не непрерывна. Например, если тариф перевозки по плечу превосходит некоторое значение, то грузопоток может полностью переключиться на другое плечо.

Для автоматизации этих исследований мы используем стохастический подход следующим образом:

- эксперт выбирает «ключевые» узлы или плечи транспортной сети и задаёт для них интервалы варьируемых параметров – тарифов обработки или пропускную способность;
- кроме этого, эксперт указывает количество вариантов, которые нужно просчитать;
- система генерирует и рассчитывает указанное количество вариантов, равномерно распределяя значения варьируемых параметров в указанных интервалах;
- эксперт исследует результаты расчётов с помощью предоставляемых системой визуальных средств и методов статистического анализа.

Естественным расширением пространственной визуализации единичного эксперимента является отображение не конкретных объёмов перевозки и загруженности, а некоторых статистических данных, собранных по всему множеству проведённых экспериментов, например, средних значений.

Пусть s – некоторое решение. Тогда набор статистических характеристик, которые поддерживает система МИКС-ПРОСТОР, можно записать в следующем виде.

- Средний объём продуктов, перевозимых по плечу (t, r, r'):
$$\underset{s}{\mathbf{avg}} \sum_p [\boldsymbol{W}^{\boldsymbol{p}}_{\boldsymbol{trr'}}]^{\boldsymbol{s}}.$$
- Средняя суммарная цена продуктов, перевозимых по данному плечу, в пересчете на один км:
$$\underset{s}{\mathbf{avg}} \sum_{p} [\boldsymbol{\varphi}^{\boldsymbol{p}}_{\boldsymbol{trr'}} \boldsymbol{W}^{\boldsymbol{p}}_{\boldsymbol{trr'}}]^{\boldsymbol{s}}.$$
- Средняя загруженность плеча как среднее отношение объемов, перевозимых по плечу, к его пропускной способности:
$$\underset{s}{\mathbf{avg}} \frac{\sum_p [\boldsymbol{W}^{\boldsymbol{p}}_{\boldsymbol{trr'}}]^{\boldsymbol{s}}}{[\boldsymbol{P}_{\boldsymbol{trr'}}]^{\boldsymbol{s}}}.$$

Для каждой характеристики, кроме собственно значения, определяющего толщину линии, можно определить границы изменения цвета линии. Например, большая загруженность представляется красными линиями, а малая – синими (рис. 4).

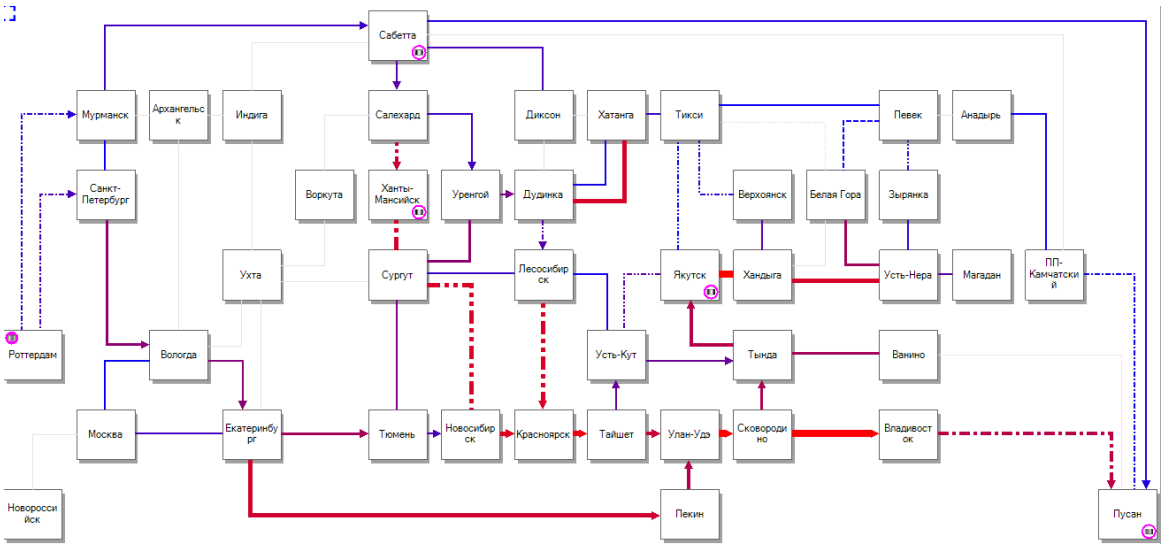


Рис. 4. Средняя загруженность плеч

Кроме перечисленных статистических характеристик, система предоставляет возможность отобразить процент решений, в которых данное плечо используется для перевозки данного продукта. На рис. 5 показан процент использования плеч для контейнеров из Роттердама. Серым цветом окрашены плечи, которые используются в большем количестве решений, а красным – в меньшем.

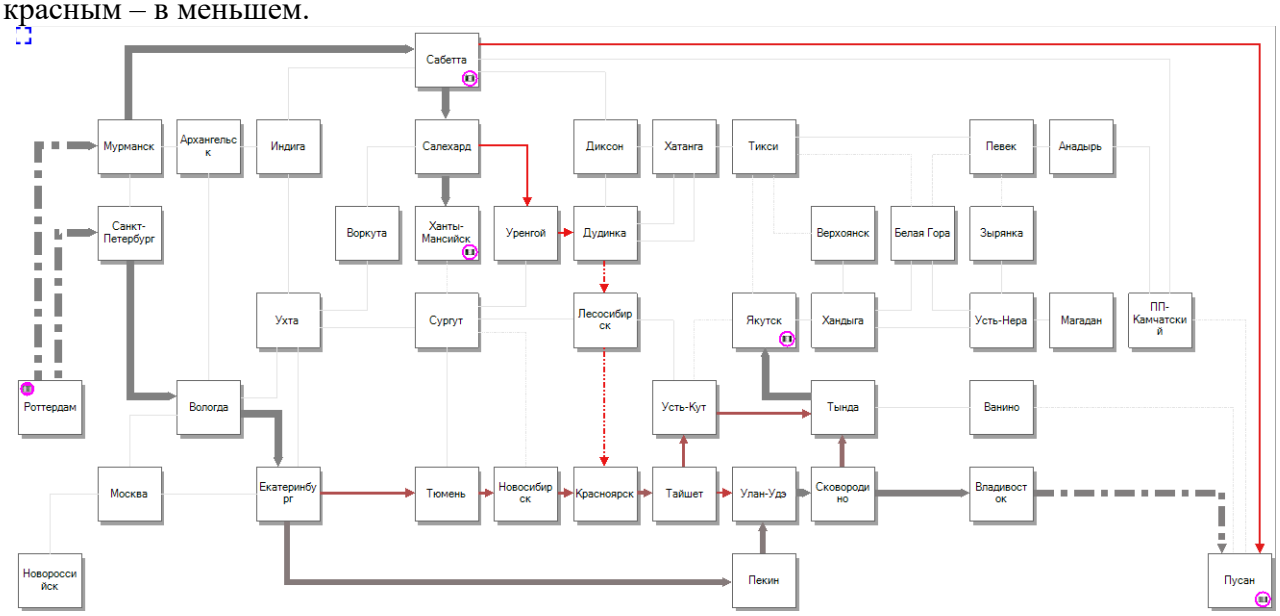


Рис. 5. Пространственное отображение процента использования плеч

Таким же образом можно отобразить и другие статистические характеристики, например, зависимость объёма перевозки по плечу от данного варьируемого параметра. Если варьируемым параметром является, скажем, тариф перевозки данного продукта по конкретному плечу, то интересующая нас зависимость может быть вычислена как линейный коэффициент корреляции Пирсона [8]:

$$\frac{\sum_s([\boldsymbol{X}]^s - \overline{\boldsymbol{X}})([\boldsymbol{Y}]^s - \overline{\boldsymbol{Y}})}{\sqrt{\sum_s([\boldsymbol{X}]^s - \overline{\boldsymbol{X}})^2 \sum_s([\boldsymbol{Y}]^s - \overline{\boldsymbol{Y}})^2}}.$$

Здесь $\overline{\boldsymbol{X}} = \underset{s}{\mathbf{avg}}[\boldsymbol{X}]^s$, $\overline{\boldsymbol{Y}} = \underset{s}{\mathbf{avg}}[\boldsymbol{Y}]^s$, а в качестве $[\boldsymbol{X}]^s$ и $[\boldsymbol{Y}]^s$ выступают значения $[\boldsymbol{\varphi}^{\boldsymbol{p}}_{\boldsymbol{trr'}}]^s$ и $[\boldsymbol{W}^{\boldsymbol{p}}_{\boldsymbol{trr'}}]^s$ соответственно.

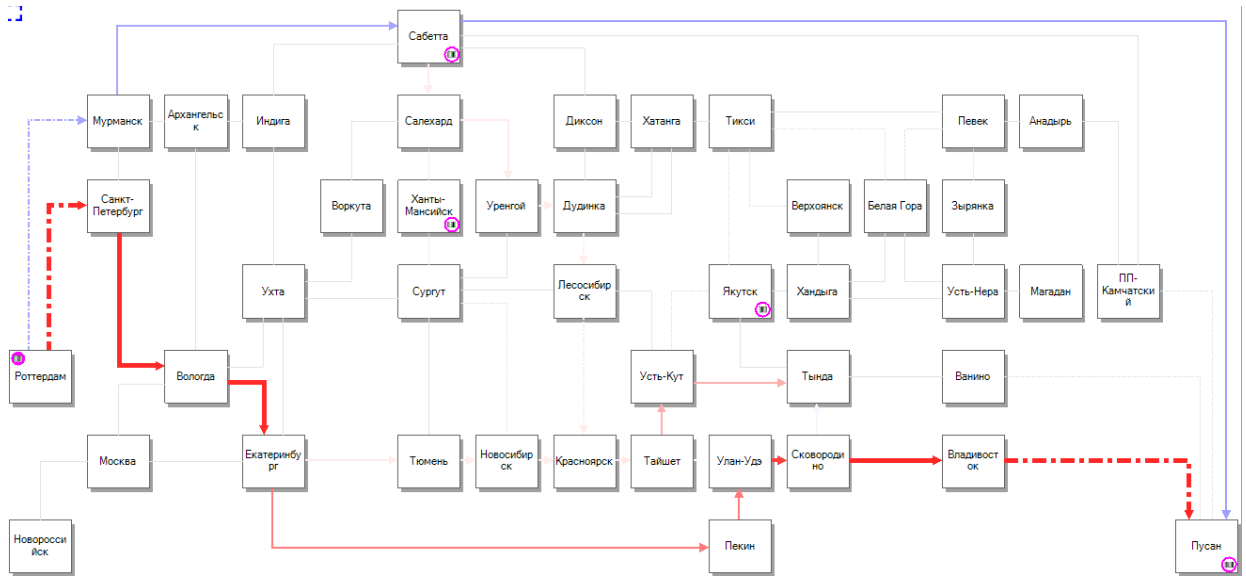


Рис. 6. Пространственное отображение корреляции

Таким образом, пользователь сразу получает представление о том, на что влияет данный параметр. На рис. 6 приведёт пример такого отображения – зависимость объёма перевозки от тарифа по плечу «Сабетта – Пусан» для контейнеров из Роттердама. Положительная корреляция представлена красными линиями, а отрицательная – синими. На основе анализа данного представления можно сделать неочевидные выводы о том, что объём перевозок контейнеров по железной дороге «Усть-Кут – Тында» возрастает при повышении варьируемого тарифа, а, например, плечо «Тында – Ванино» от этого тарифа существенно не зависит.

Рассматривая объёмы перевозки по плечам как характеристики конкретного решения, можно ставить вопрос о схожести вариантов и, следовательно, ставить задачу кластеризации множества рассчитанных вариантов. Вопрос о качестве кластеризации и выборе наиболее подходящего метода – тема отдельного исследования.

Имея разбиение множества вариантов решений на кластеры, можно выделить их наиболее типичных представителей, а также показать частотные характеристики распределения значений варьируемых параметров в каждом кластере. Пример такого отображения представлен на рис. 7, из которого видно, что для выделенного кластера характерно малое значение по первым двум варьируемым параметрам.

#	Решен в класте	Екатеринбург - Пекин (Жд) - Пр.Способность	Новосибирск - Красноярск (Жд) - Пр.Способность	Сабетта - Пусан (Лед) - Тариф	Суммарная цена перевозки
1	92	48.433	74.009	1.198	545674.779
2	54	64.367	27.329	1.277	566007.129
3	32	55.315	53.629	0.123	515273.211
4	11	46.325	9.385	0.879	600959.710
5	5	5.999	25.350	1.375	594401.505
6	3	1.531	4.637	1.520	665115.386
7	3	56.128	42.855	0.858	547688.756

Рис. 7. Кластеризация множества решений

Транспонирование характеристической матрицы позволяет вычислять корреляцию не между вариантами решений, а между плечами. Знание этой корреляции даёт возможность выделения транспортных коридоров, которые представляют собой некоторую совокупность плеч с определённой степенью вероятности либо одновременно задействованных, либо одновременно не задействованных в перевозке. Следует отметить, что для разных продуктов коридоры могут отличаться. Если в матрице корреляции раскрасить клетки, варьируя цвет от красного (для положительной корреляции) до синего (для отрицательной корреляции), а затем переставить строки и столбцы матрицы так, чтобы красный цвет сосредоточился около главной диагонали, то возникающие красные блоки и будут означать транспортные коридоры. Пример такого отображения показан на рис. 8.

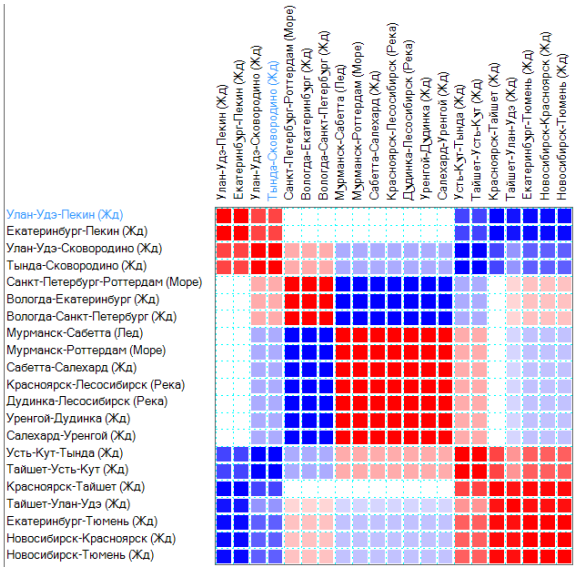
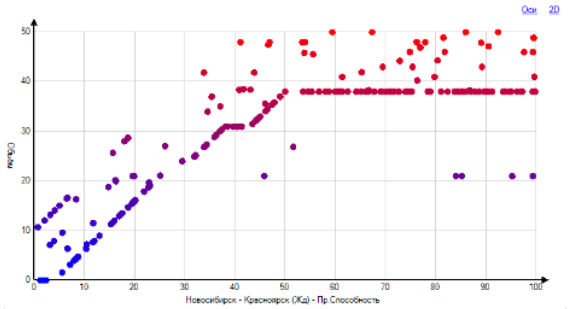
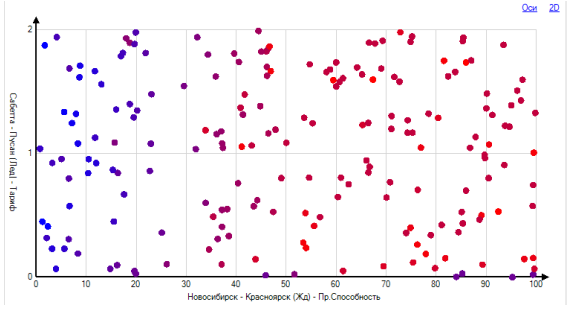


Рис. 8. Корреляция плеч – транспортные коридоры

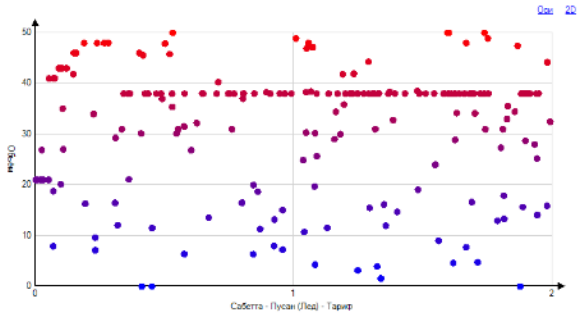
Если варьируемых параметров несколько, то естественным образом возникает необходимость многомерной визуализации. Пусть, например, эксперт одновременно меняет тарифы для двух плеч. Отдельно 2D-графики зависимости объёма перевозки по некоторому третьему плечу могут не давать адекватного полного представления (рис. 9а и 9б). Оставаясь в двумерном пространстве, мы можем отображать объём цветом и размером точек на плоскости, каждая из которых соответствует конкретному варианту, как показано на рис. 9в. Такой способ визуализации можно назвать 2D+.



(а)



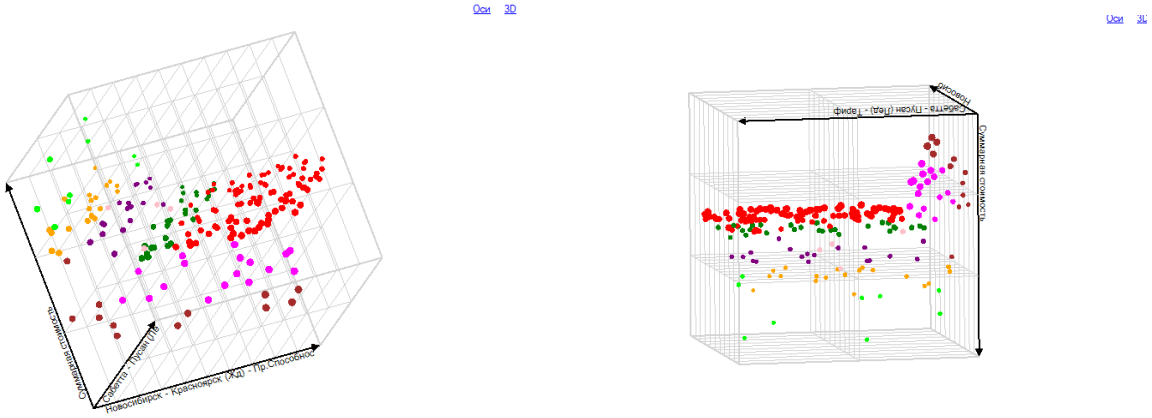
(б)



(в)

Рис. 9. Двумерная визуализация

Возможна также и 3D-, и 3D+-визуализация, причём осям могут соответствовать как значения варьируемых параметров, так и любые вычисляемые числовые величины, а отображение отдельных точек не обязательно должно напрямую определяться перевозимым объёмом. Так, на рис. 10 осям соответствуют два варьируемых параметра и суммарная стоимость перевозки, а цвет точки определяется цветом кластера, к которому отнесён данный вариант. Естественно, что в трёхмерном случае необходима возможность вращать изображение в пространстве.



(а) (б)

Рис. 10. Трёхмерная визуализация

На этом примере (рис. 10б) становится очевидным предположение, что разбиение на кластеры при достаточно больших значениях тарифа «Сабетта – Пусан» (горизонтальная ось) в основном связано с суммарной стоимостью перевозки (вертикальная ось), а при малых – со значением этого тарифа.

Заключение

Мы рассмотрели средства автоматизации прогнозирования развития опорной транспортной сети, предоставляемые системой МИКС-ПРОСТОР. Можно заметить, что по мере развития системы разработка новых средств визуализации позволяет не столько облегчить восприятие исследователем/экспертом полученного решения, сколько осознать и сформулировать запросы на новые средства анализа, которые, в свою очередь, требуют новых средств визуализации и т.д.

Литература

1. *Бульонков М. А., Карпан В. В., Малов В. Ю., Марусин В. В., Радченко В. В.* Концептуальные вопросы построения Модельно-Информационно-Картографической Системы (МИКС) //

Моделирование производственных и региональных систем на основе ГИС и информационных технологий: сб. науч. тр. Новосибирск: ИЭОПП СО РАН, 2011. С. 5–28.
2. *Бульонков М. А., Филаткина Н. Н.* Ситуационный анализ в системе транспортного прогнозирования МИКС-ПРОСТОР // Информационные технологии. 2013. № 8. С. 43–52.
3. *Воробьёва В. В., Малов В. Ю., Радченко В. В., Поттер М. В., Серебрянников И. Е.* Модель прогнозирования развития опорной транспортной сети // Моделирование производственных и региональных систем на основе ГИС и информационных технологий: сб. науч. тр. Новосибирск: ИЭОПП СО РАН, 2011. С. 68–96.
4. *Гранберг А. Г.* Оптимизация территориальных пропорций народного хозяйства. М.: Экономика, 1973.
5. Азиатская часть России: моделирование экономического развития в контексте опыта истории / под ред. В. А. Ламина, В. Ю. Малова. Новосибирск: Изд-во СО РАН, 2012. 463 с.
6. *Забиняко Г. И.* Пакет программ целочисленного линейного программирования // Дискретный анализ и исследование операций. Сер. 2. 1999. Т. 6, № 2. С. 32–41.
7. Google OR-Tools [Электронный ресурс]. URL: `https://developers.google.com/optimization/` (дата обращения: 22.03.2019).
8. *Гмурман В. Е.* Теория вероятностей и математическая статистика: учебное пособие для вузов. 10-е издание, стереотипное. М.: Высшая школа, 2004. 479 с.

Статья поступила в редакцию 31.07.2019;
переработанный вариант – 30.08.2019.

Бульонков Михаил Алексеевич
к.ф.-м.н., доцент кафедры программирования НГУ, зав. лаб. смешанных вычислений ИСИ СО РАН, тел. (383) 330-93-44, e-mail: `mike@iis.nsk.su`.

Нестеренко Татьяна Викторовна
доцент кафедры систем информатики НГУ, н.с. ИСИ СО РАН (630090, Новосибирск, пр. Лаврентьева, 6), e-mail: `nest@iis.nsk.su`.

Research automation of the transport network development

M. Bulyonkov, T. Nesterenko

A research automation system of forecasting the development of a basic transport network in Russia is considered. A formal model of the transport network allows various types of transport, products, production and consumption, and the task is to minimize the total cost of transportation with respect to a set of linear restrictions. The system provides a user interface for setting and editing all parameters of the transport network. Particular attention is paid to the visual and interactive presentation of simulation results. A holistic perception of the results is achieved by displaying both the input data (mode of transport, throughput, transportation cost) and the modeling results (transported volume and functioning capacity) directly either on the map or on the network plan. The system allows us to display either a selected product or all products together. In practice, it is often required to solve the problem that is inverse to simulation, for example, to find out at what tariffs the volumes of transportation over a certain transportation shoulder will exceed a given value. To do this, it is proposed to use stochastic methods based on the mass solution of the transport problem for a variety of network parameters, such as the capacity of the transport shoulder, the tariff for transportation of cargo or its processing in the transport hub. Clustering methods make it possible to distinguish a relatively small number of "typical" solutions from the entire set of variants. This enables the expert to evaluate both the conditions and the probability of the predicted transport situation. It is also shown that this approach allows us to identify automatically emerging transport corridors and determine the dependence of the volume of transportation on a given shoulder from a specific variable parameter.

Keywords: transportation problem, modeling, research automation system.

УДК 519.632

Управление посредством семантической сети прикладным программным комплексом для решения задач математической физики

Л. А. Голубева, В. С. Горшунов, В. П. Ильин

Численное решение задач математической физики с использованием ЭВМ можно разбить на несколько этапов: построение геометрической модели расчётной области, построение сеточной модели, аппроксимация функций, производных и интегралов, а также решение уравнений. Существует множество сеточных генераторов и алгоритмов для построения двумерных и трехмерных сеток, программ для решения систем уравнений, аппроксиматоров, средств геометрического моделирования. При создании прикладного программного комплекса для решения задач математической физики, базирующегося на концепции базовой системы моделирования, каждый из этапов решения задачи можно представить в виде отдельного модуля. Каждый модуль, в свою очередь, может представлять из себя набор алгоритмов и подпрограмм. Такой программный комплекс обеспечивает целостность решения вычислительной задачи благодаря широкому набору инструментов для прохождения любого из вычислительных этапов и позволяет варьировать входные параметры, выбирать наиболее подходящие алгоритмы на разных этапах. Более того, такая система позволяет осуществлять декомпозицию исходной расчётной области на подобласти при построении геометрии и генерацию квазиструктурированной сеточной модели. Однако при включении новых алгоритмов и программ в вычислительный комплекс неизбежно возрастает сложность его использования. Таким образом, возникает потребность в проектировании надсистемы, которая позволит определить наилучший с точки зрения некоторых критериев качества, определённых заранее, набор алгоритмов решения подзадач на каждом из этапов. Цель настоящей работы – разработать и описать такую модель управления данным вычислительным комплексом с помощью базы знаний, представленной в виде семантической сети.

Ключевые слова: математическая физика, геометрическая структура данных, сеточная структура данных, функциональная структура данных, БСМ, семантические сети, базы знаний.

1. Введение

База знаний (БЗ) – это особого рода база данных, разработанная для оперирования знаниями (метаданными). База знаний содержит структурированную информацию, покрывающую некоторую область знаний. Современные базы знаний работают совместно с системами поиска информации, имеют классификационную структуру и формат представления знаний.

Представление знаний – вопрос, связанный с подбором представления конкретных и обобщённых знаний, сведений и фактов для накопления и обработки информации в ЭВМ. Основная проблема – научиться хранить знания таким образом, чтобы программы могли эффективно обрабатывать их. Для представления знаний можно использовать семантические сети [1]. Каждый *узел* такой сети представляет часть вычислительного алгоритма, а *дуги* используются для определения отношений между узлами.

Одна из проблем в представлении знаний, связанная с искусственным интеллектом (ИИ) – хранение и обработка знаний в информационных системах. Примеры применения ИИ – экспертные системы, машинный перевод, компьютеризированное техническое обслуживание и системы извлечения и поиска информации, пользовательские интерфейсы баз данных [2].

В настоящей статье описывается подход к построению базы знаний для управления программным комплексом для решения 3-мерных задач математической физики на основе концепции базовой системы моделирования (БСМ) [3].

2. Описание семантической сети, моделирующей управление вычислительным процессом

При решении задач математической физики можно выделить три этапа: построение геометрической модели, построение сеточной модели, аппроксимация и решение уравнений [4]. Составим семантическую сеть, которая будет включать в себя в качестве подсетей 3 семейства узлов, связанных последовательно, каждое из которых ответственно за один из следующих этапов вычислительного процесса:

1. «Вороной» – задачи геометрического и функционального моделирования.
2. «Делоне» [5] – генерация сеток.
3. «Чебышёв» [6] – аппроксимация исходных уравнений.

Каждая из трёх подсетей, в свою очередь, также является семантической сетью. Связи между узлами в этой семантической сети предоставляют возможность организовать конкретный вычислительный процесс. На каждом из трёх этапов решения задачи, представленных в виде подсети, возможны различные варианты развития событий. Модель управления предполагает построение пути в сети, который представляет вычислительный процесс. Прохождение по этому пути гарантирует решение задачи. В зависимости от того, какие условия накладываются на качество решения на каждом из этапов, мы можем либо принять такой подход, либо выбрать иной путь построения решения в любой из трёх подсетей.

3. Описание узлов семантической сети

Каждый узел в семантической сети представляет определённый алгоритм. Узлы в сети могут быть одного из трёх типов в зависимости от того, к какой подсети они принадлежат. От типа узла зависит набор входных и выходных данных.

Все три подсети последовательно связаны между собой: «Вороной» – «Делоне» – «Чебышёв». В свою очередь, узлы внутри каждой из подсетей тоже могут между собой быть связаны, такое утверждение основывается на принципе декомпозиции расчётной области на подобласти. Цель декомпозиции – использование наиболее подходящего алгоритма генерации сетки в каждой из подобластей с точки зрения качества получаемого решения задачи во всей области. Опишем каждую из трёх подсетей.

3.1. «Вороной»

Подсеть «Вороной» ответственна за построение и визуализацию геометрии, а также за формирование геометрической структуры данных (ГСД) и функциональной структуры данных (ФСД) [7]. Входные данные для узлов подсети – описание расчётной области для конкретной решаемой задачи, которое может быть задано при помощи диалогового окна либо в виде текста на соответствующем языке описания геометрии. Выходные данные – различные ФСД, ГСД и их графическое представление. ФСД включает в себя данные, необходимые для

функционирования узлов других подсетей, например, граничные значения, начальные разбиения границ для построения сеток и т.д.

Узлы могут иметь различное представление ГСД. Однако для каждой из ГСД можно построить общее внутреннее представление на основе какой-либо платформы с возможностью визуализации геометрии. В качестве такой платформы в настоящей работе используется «Гербарий» [8] – инженерно-исследовательская программная платформа. В «Гербарии» реализована возможность построения и визуализации геометрии. Геометрическая модель в «Гербарии» представляет собой набор примитивов, к которым применены булевы операции объединения, вычитания. Под набором примитивов в данном случае понимаются объёмные фигуры, полученные путём выдавливания или вращения криволинейной плоской замкнутой линии.

3.2. «Делоне»

Задачей узлов этой подсети ставится построение сеточной модели. Для каждого такого узла существует определённый набор входных значений – некоторые ГСД и ФСД, сформированные в подсети «Вороной», и набор выходных данных – сеточная структура данных (ССД) [7].

ССД, которая представляет сеточную модель, может включать в себя набор сеточных вершин, сеточных рёбер, сеточных граней и т.д. ССД каждого из узлов может различаться, но также, как и в случае с ГСД, может быть сведена к некоторой общей ССД, а значит, и иметь одно графическое представление на выбранной платформе, в нашем случае в «Гербарии».

Также сеточная модель обладает различными свойствами: качество сетки, структурированность, адаптивность [7]. Качественными характеристиками могут служить максимальный и минимальный угол по всем элементам, их отношение; максимальная и минимальная длина ребра; радиус вписанной или описанной окружности. В зависимости от условий, накладываемых на качество, можно выбрать тот или иной путь при управлении вычислительным процессом.

3.3. «Чебышёв»

Целью функционирования подсети является аппроксимация исходных уравнений и решение исходной задачи. Решение задачи будем называть алгебраической структурой данных (АСД) [9]. Входными данными служат результаты, полученные на предыдущих этапах (в предыдущих подсетях): ССД и ФСД.

В зависимости от постановки исходной задачи её решением может быть решение некоторой системы дифференциальных, интегральных или линейных алгебраических уравнений в узлах или ячейках сеточной модели [10].

3.4. Пример семантической сети

На каждом этапе решения исходной задачи можно ввести критерии аварийной остановки, когда результат прохождения этапа неудовлетворителен и необходимо выбрать иной путь (см. рис. 1). Так, при декомпозиции геометрической модели на подобласти последние могут быть описаны в несовместимых форматах. Важно отметить, что такой случай может возникнуть при построении геометрии для генерации на ней квазиструктурированной сетки [7]. В качестве такого критерия остановки для работы подсети «Делоне» можно рассматривать качественные характеристики сетки. От подсети «Чебышёв» можно требовать соответствия решения уравнений определённым условиям. Каждый способ задания геометрии, построения сетки, аппроксимации и решения уравнений – это отдельный узел сети.

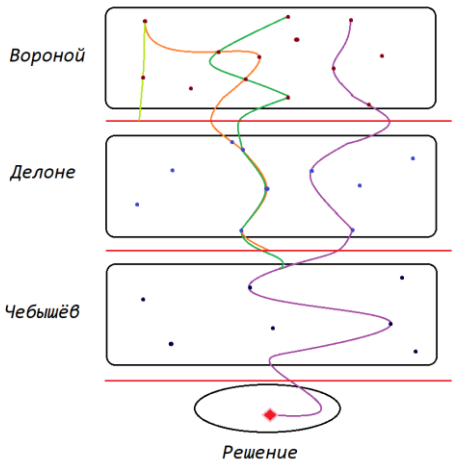


Рис. 1. Визуальное представление сети

Несмотря на то, что путь в некоторых подсетях может совпадать, итоговое решение задачи может различаться с точки зрения качества.

4. Использование БЗ для управления вычислительным комплексом

Опишем ГСД, ФСД, необходимые для функционирования некоторых сеточных генераторов, а также ССД, полученные в результате работы таких генераторов, и приведём пример использования описанной концепции.

CubDat – сеточный генератор, способный строить структурированные сетки с равномерным шагом в трёхмерных областях, состоящих из параллелепипедов [11].

Базовые координаты (БК) – упорядоченный набор уникальных значений координат для каждой из координатных осей.

ГСД и ФСД CubDat выглядит следующим образом:

nx, *ny*, *nz* – число БК для каждой из осей координат;

$k_1, k_2, \ldots, k_{nk}$ – список БК по каждой из трёх осей. Здесь и далее $k = x \mid y \mid z$;

$sk_1, sk_2, \ldots, sk_{nk-1}$ – список числа шагов разбиения между базовыми координатами;

sk_i – задает число шагов разбиения между БК k_i и k_{i+1};

nmat – число материалов в расчетной области, не может быть меньше 1;

nsd – число подобластей (параллелепипедов) в расчетной области, не может быть меньше 1;

$sd_1.t$ – номер материала подобласти;

$sd_1.p$ – номер приоритета подобласти;

$sd_1.kb$, $sd_1.ke$ – номера БК оси Ok начала и конца подобласти соответственно, нумерация ведется с 0;

$sd_1.bc_{kb}$, $sd_1.bc_{ke}$ – номера граничных условий на гранях подобласти $k = k_{sd1.kb}$ и $k = k_{sd1.ke}$ соответственно;

$ndbc_i$ – номер значения условия Дирихле, соответствующий узлу сетки;

n_dbc – число значений условий Дирихле;

$Vdbc_i$ – значения условия Дирихле.

В ССД CubDat содержится описание узлов:

nnode – число узлов в расчетной области;

x_i, y_i, z_i – декартовы координаты узлов;

ntet – число тетраэдров в расчетной области;

n_i, n_j, n_k, n_l – номера узлов – вершин тетраэдра, нумерация узлов идет с 0;

bc_i, bc_j, bc_k, bc_l – номера граничных условий на гранях тетраэдра.

NetGen – универсальный сеточный генератор. Входные данные для NetGen должны быть записаны в формате .stl, .geo, .step. Выходной формат данных – .vol. Описание этих форматов доступно в открытых источниках [12].

В программном комплексе [3] посредством «Гербария» формируется внутренняя ГСД, входные данные для которой задаются с использованием графического интерфейса в виде диалогового окна. ГСД «Гербария» может быть преобразована на уровне программного кода в формат, понятный другим узлам семантической сети: CubDat и NetGen. Более того, соответствующая ГСД будет иметь визуальное представление в Демонстраторе «Гербария». То же справедливо и для ССД.

4.1. Задание геометрии, формирование входных данных и ФСД для сеточного генератора CubDat. Модуль «Вороной»

Создадим для примера геометрическую модель, представленную в виде двух тел кубической формы, указав начальные и конечные координаты диагонали каждого из тел (рис. 2).

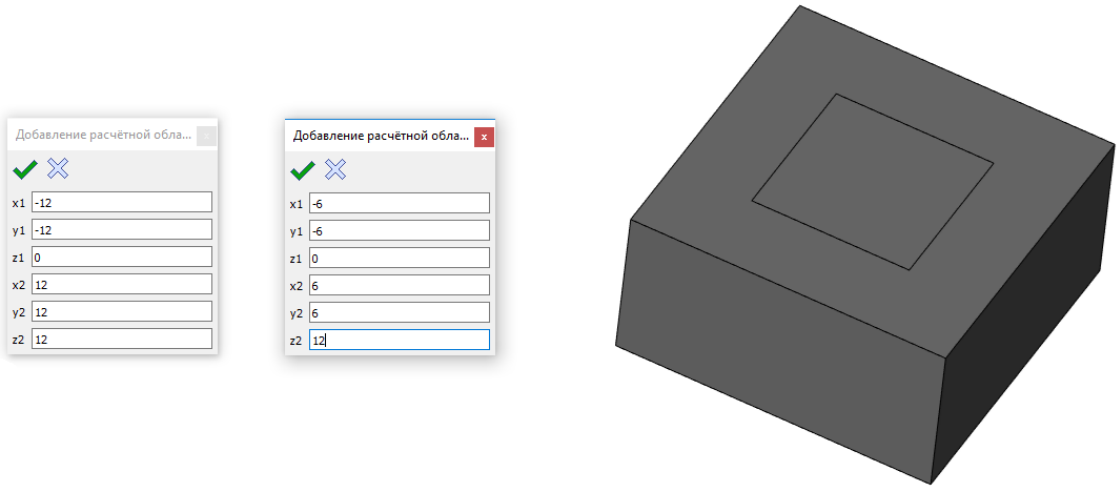


Рис. 2. Задание геометрии

Зададим отношение между телами с помощью булевых операций, ФСД и параметры генерации сетки (рис. 3).

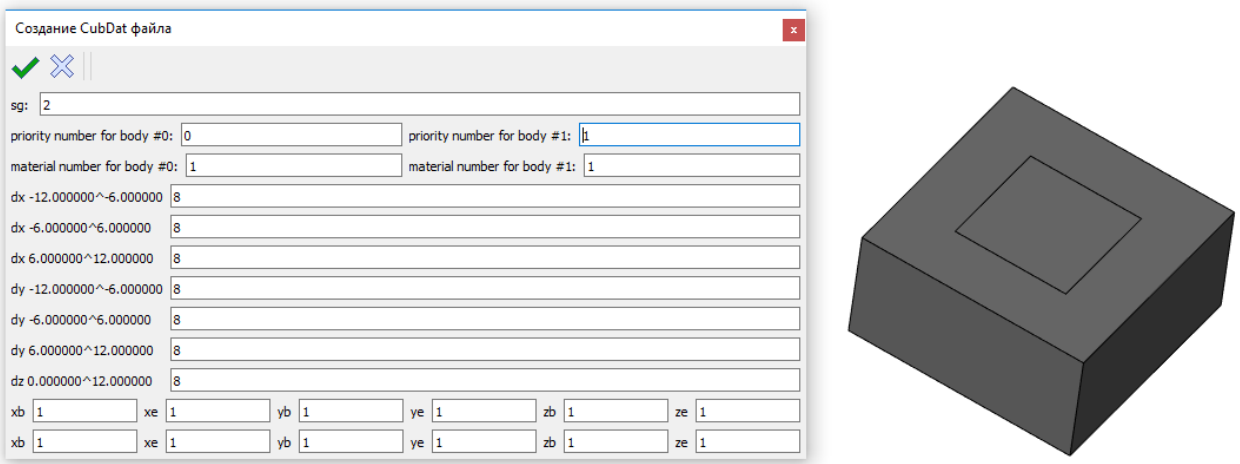


Рис. 3. Параметры генерации сетки

4.2. Генерация сетки. Модуль «Делоне»

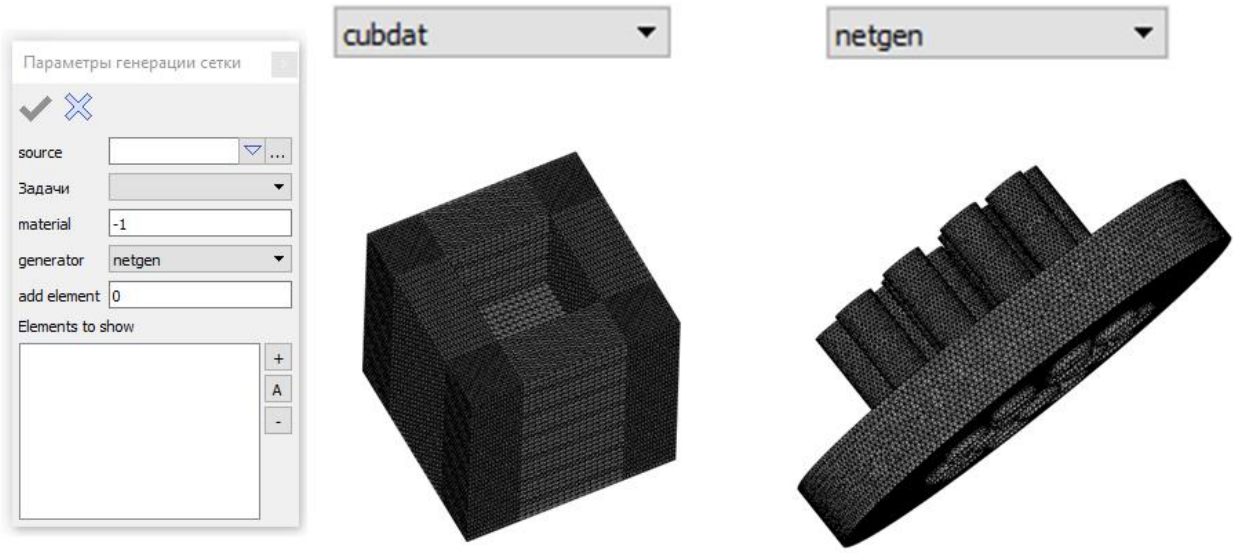


Рис. 4. Сеточные генераторы

Указав источник генерации сетки, получим визуальное представление построенной сеточной модели (рис. 4). При формировании квазиструктурированных сеток [7] (рис. 5) необходимо указывать свой источник для каждой из подобластей, выбирая соответствующие подобласти посредством ниспадающего списка «Задачи». Далее вызывается программа для аппроксимации уравнений и построения решения (Модуль «Чебышёв»). Визуализация полученного решения находится в разработке.

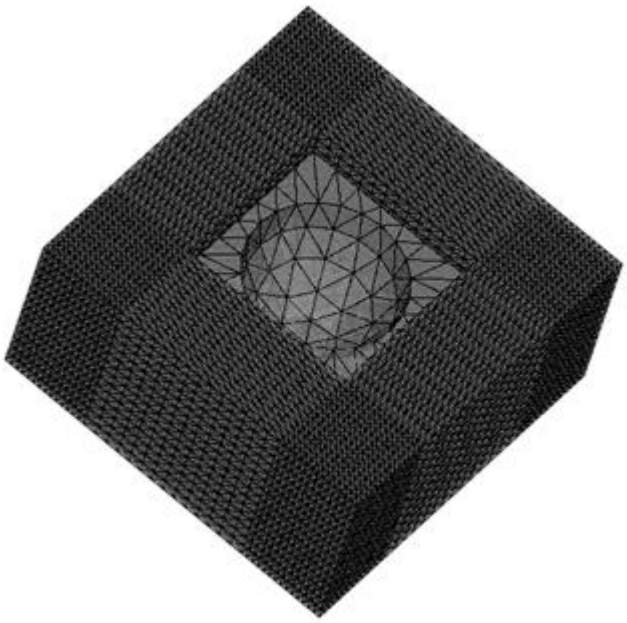


Рис. 5. Квазиструктурированная сетка

5. Заключение

В настоящей работе представлена модель управления вычислительным комплексом на основе семантической сети. Узлы этой сети описаны и классифицированы по трём типам семейств: «Вороной», «Делоне», «Чебышёв». Такой способ управления опробован на программном комплексе для решения 3-мерных задач математической физики на основе концепции БСМ [7]. В дальнейшем предполагается расширение этой сети путём добавления новых узлов в каждую подсеть.

Литература

1 *Загорулько Ю. А., Загорулько Г. Б.* Инженерия знаний: учебное пособие. Новосибирск: РИЦНГУ, 2016. 93 с.

2 Basic Research Needs Workshop for Scientific Machine Learning Core Technologies for Artificial Intelligence // Prepared for Department of Energy Advanced Scientific Computing Research. February 10, 2019.

3 *Ильин В. П., Гладких В. С.* Базовая система моделирования (БСМ): концепция, архитектура и методология // Тр. конф. «Современные проблемы математического моделирования, обработки изображений и параллельных вычислений» (СПММОИиПВ), Ростов-на-Дону, ДГТУ, 2017. С. 151–158.

4 *Ильин В. П.* Фундаментальные вопросы математического моделирования // Вестник Российской Академии Наук. 2016. Т. 86, № 4. С. 26–36.

5 *Ильин В. П.* DELAUNAY: технологическая среда генерации сеток // СибЖИМ. 2013. Т. 16. С. 83–97.

6 *Бутюгин Д. С., Ильин В. П.* CHEBYSHEV: принципы автоматизации построения алгоритмов в интегрированной среде для сеточных аппроксимаций начально-краевых задач // Труды Международной конференции ПАВТ’2014, Челябинск, ЮУрГУ, 2014. С. 42–50.

7 *Голубева Л. А., Горшунов В. С., Ильин В. П., Эрдыниев Э. Б.* Программный комплекс для решения 3-мерных задач математической физики на основе концепции БСМ // Труды Международной конференции «Вычислительная математика и математическая геофизика» к 90-летию со дня рождения академика А. С. Алексеева, 2018. С. 126–132.

8 «ГЕРБАРИЙ» [Электронный ресурс]. URL: `http://tflex.ru/about/publications/detail/index.php?ID=3846` (дата обращения: 30.06.2019).

9 *Бутюгин Д. С., Гурьева Я. Л., Ильин В. П., Перевозкин Д. В., Петухов А. В., Скопин И. Н.* Функциональность и технологии алгебраических решателей в библиотеке Krylov // Вестник ЮУрГУ. Сер. «Вычислительная математика и информатика». 2013. Т. 2, № 3. С. 92–105.

10 *Ильин В. П.* Математическое моделирование: Ч. 1: Непрерывные и дискретные модели. Новосибирск: Изд. СО РАН, 2017. 429 с.

11 *Gurieva Ya. L., Il’in V. P.* Program package for 3D boundary-value elliptic Problem // Bull. NCC, series: “Num. Anal.”. 2002. Iss. 11. P. 35–52.

12 NETGEN [Электронный ресурс]. URL: `https://ngsolve.org/docu/latest/` (дата обращения: 30.06.2019).

Статья поступила в редакцию 05.07.2019;
переработанный вариант – 09.08.2019.

Голубева Людмила Андреевна

к.ф.-м.н., доцент, научный сотрудник Института вычислительной математики и математической геофизики СО РАН (630090, Новосибирск, просп. Ак. Лаврентьева, 6), e-mail: golubeva@labchem.sscc.ru.

Горшунов Василий Сергеевич

инженер Института вычислительной математики и математической геофизики СО РАН, e-mail: basil.gorshunov@gmail.com.

Ильин Валерий Павлович

д.ф.-м.н., профессор, главный научный сотрудник Института вычислительной математики и математической геофизики СО РАН, e-mail: ilin@sscc.ru.

Management of an application software package for solving problems of mathematical physics via semantic network

L. A. Golubeva, V. S. Gorshunov, V. P. Il'in

The numerical solution of mathematical physics problems using a computer can be divided into several stages: geometric model of the computational domain construction, grid model construction, function approximation, derivatives and integrals, as well as equations solving. There are many grid generators and algorithms for constructing two-dimensional and three-dimensional grids, programs for systems of equations solving, approximators and geometric modeling tools. While creating an application software package for solving problems of mathematical physics based on the concept of basic modeling system, each stage of the problem solving representing as a separate module. Each module, in turn, can be a set of algorithms and subroutines. Such a software package ensures the integrity of the solution to a computational problem thanks to a wide range of tools for going through any of the computational stages and allows you to vary the input parameters and choose the most suitable algorithms at different stages. Moreover, such a system allows the decomposition of the initial computational domain into subdomains when constructing the geometry and the generation of a quasistructured grid model. However, when new algorithms and programs are included into the computing complex, the complexity of its use increases inevitably. Thus, there is a need for the design of a supersystem allowing us to determine the best set of algorithms for subtasks solving at each stage in terms of some quality criteria defined in advance. The purpose of this work is to develop and describe such a model for managing this computing complex using a knowledge base presented in the form of semantic network.

Keywords: mathematical physics, geometric data structure, grid data structure, functional data structure, BMS, semantic networks, knowledge bases.

УДК 004.942

Моделирование агентного окружения при разработке мультиагентной системы на примере крупномасштабных инфраструктурных проектов

Т. Н. Есикова[1], С. В. Вахрушева

Мультиагентное моделирование позволяет отразить неоднородность, уникальность, многообразие и динамику взаимодействия конкретных экономических акторов, а значит, и структуру моделируемого процесса наиболее приближенно к реальности. Это обусловило выбор мультиагентного подхода в качестве инструмента имитационного моделирования процесса реализации крупномасштабного инфраструктурного проекта на примере ТКМ (трансконтинентальной магистрали) через Берингов пролив на территориях Азиатской России. В статье изложены основные идеи формирования ключевой части мультиагентной системы – окружения агентов (информационного пространства) как аналитической основы принятия решений акторами различной природы (экономическими, управленческими и др.).

Ключевые слова: мегапроекты, мультиагентные системы (МАС), агенты, среда взаимодействия агентов, архитектура МАС, оценка последствий.

1. Введение

Высокий уровень неопределенности экономических и экологических результатов, которые заявляются инициаторами крупномасштабных инфраструктурных проектов, влечёт за собой необходимость моделирования данных процессов. Кроме того, основные факторы риска связаны с пересечением интересов экономических акторов различной природы (в том числе социальных групп), имеющих разный вес и обладающих разными уровнями влияния при принятии окончательных решений. В связи с этим представляет особый интерес такой инструмент имитационного моделирования, как мультиагентный подход [1–8], который позволяет не только детально представить каждого экономического актора (с его интересами, механизмами влияния на управленческие решения, особенностями поведения и др.), но и саму гетерогенную среду окружения.

Особое место в этом контексте занимает моделирование информационного пространства [9], предопределяющего аналитическую основу формирования решений акторами разной природы (экономическими, управленческими и др.). Это предопределило необходимость отдельного рассмотрения данного этапа проектирования мультиагентной системы для имитации различных вариантов реализации инфраструктурных мегапроектов.

Поскольку речь идёт о моделировании системы с обратной связью, следует подробнее рассмотреть концепцию взаимодействия агента и среды его окружения. Идея, лежащая в основе информационного взаимодействия, состоит в том, что агенты адаптируют свое поведение в зависимости от окружения. Таким образом, осуществление этой связи дает информацию о причине и следствии происходящих в системе событий, о последствиях действий агентов и о том, какую стратегию следует выбрать агентам для достижения их целей.

[1] Работа выполнена по плану НИР ИЭиОПП СО РАН № АААА-А17-117022250123-0.

На данном этапе исследования была поставлена цель – охватить наиболее важные аспекты реальной проблемы, когда агент взаимодействует со своей средой.

2. Модель разрабатываемой МАС

Специфика выбранного метода исследования предопределяет идентификацию представительного множества акторов в качестве агентов и декомпозицию общего процесса реализации проекта на ключевые компоненты [1].

В общем случае модель разрабатываемой МАС имеет вид [2]:

$M = (V, E, I, P, Q)$, где:

V – множество различных групп агентов. На текущем этапе исследования выделены следующие типы агентов: представители высших органов власти, руководители территориальных субъектов в зоне реализации проекта, эксперты, инвесторы, подрядные организации, поставщики оборудования и общественные организации;

$E \subset V \times V$ – множество различных видов отношений между агентами (ключевых подпроцессов), например, согласование проекта ТКМ (в том числе при внесении изменений в проект), взаимодействие власти и компаний-инвесторов, проведение тендеров по отдельным этапам работ по проекту, закупка оборудования и т.д.;

$I = I_{field} \cup I_{flow}$ – окружение агентов, множество компонентов внешней информационной среды функционирования агентов – информационного пространства, которое состоит из информационных полей I_{field} и информационных потоков I_{flow}, в роли которых могут выступать, к примеру, СМИ;

$P \subset V \times I$ – множество взаимосвязей агентов и внешней среды их функционирования (восприятие агентами преобразований среды, формирование изменений в информационных потоках);

$Q \subset I \times V$ – множество взаимосвязей среды и агентов (оповещение агентов информационной средой о каком-либо событии, изменение атрибутов агентов и т.д.).

Поскольку при протекании процессов, связанных с развитием транспортной сети, информационное пространство занимает особое место (т.к. влияет на исходы взаимодействий различных участников этих процессов), рассмотрим более подробно некоторые аспекты идентификации типов агентов и проектирования окружения агентов на примере реализации крупномасштабных транспортных проектов.

3. Идентифицированные типы агентов

В результате декомпозиции процесса реализации проекта ТКМ через Берингов пролив на территориях Азиатской России были выявлены следующие группы агентов:

- представители федеральных министерств (агентства, службы, надзоры: Росжелдор, Росавтодор, Росприроднадзор и др.);
- структуры региональных и муниципальных органов власти автономного округа;
- компании-инвесторы (ресурсодобывающие компании, ведущие разработку в регионе) или отдельные инвесторы;
- инжиниринговые строительные организации и инженерные организации, непосредственные разработчики проектов;
- общественные организации (сообщества по защите прав коренных малочисленных народов Севера, общественная экологическая экспертиза, общественные организации природоохранного направления автономного округа, например, такие как «Умкы-патруль» и др.);
- компании-поставщики оборудования;
- подрядные организации;
- СМИ.

Но при детальном рассмотрении процесса формирования и изменения окружающей среды удобнее рассматривать эти группы агентов в качестве следующих:

- агенты-управленцы – $V^{\sim}$;
- агенты- лоббисты – $V^{\approx}$;
- агенты-участники информационного сопровождения проектов – V^{Δ}.

Отметим, что между агентами разных групп ($V^{\sim}, V^{\approx}, V^{\Delta}$) осуществляются взаимодействия в обоих направлениях. Концепция «многие-ко-многим» лучшим образом отражает суть процессов, которые свойственны схемам взаимодействия в реальном мире. Технически взаимодействие агентов (агентов друг с другом, агента с окружающей средой) осуществляется через передачу потоков сообщений.

Первое подмножество агентов ($v_j^{\sim} \in V^{\sim}$, $[V^{\sim}] = n_1$) включает в себя классических агентов (агенты-управленцы), функционирующих в экономическом пространстве: руководители территориальных структур, администрация проекта сооружения транспортной магистрали, инвесторы, представители населения, общественных движений и т.п.

Второе подмножество агентов ($v_j^{\approx} \in V^{\approx}, [V^{\approx}] = n_2$) состоит из агентов, которые обеспечивают лоббирование проектов на разных уровнях принятия решений, которое могут проводить как отдельные агенты-лоббисты, так и организации-лоббисты проекта.

Третье подмножество агентов ($v_j^{\Delta} \in V^{\Delta}, [V^{\Delta}] = n_3$) – это группа агентов информационного сопровождения проектов, включающая в себя как отдельные подмножества агентов, так и узкоспециализированных агентов (агенты-интерпретаторы сообщений и др.).

На этапе имитации между перечисленными группами агентов возникают взаимодействия, отражающие суть процессов, протекающих в реальности. В данном случае взаимодействие агентов (как между собой, так и с окружающей средой) технически осуществляется посредством передачи сообщений.

4. Построение концептуальной модели информационного пространства

В рамках разрабатываемой МАС окружение агентов (I) – абстрактное представление агрегации различных составляющих информационного пространства (как информационных полей, так и потоков) [10, 11]. В контексте программной системы это база данных, причём под информационными полями (I_{field}) понимаются данные, в которые не вносятся изменения агентами (базы знаний о погодных условиях региона, справочники, содержащие информацию о составе тех или иных строительных работ, информацию о существующих транспортных проектах и т.д.), а под информационными потоками (I_{flow}) – данные, которые формируют и изменяют агенты (сообщения СМИ о событиях, связанных с реализацией проекта, сообщения иностранных СМИ, упоминания о проекте в социальных медиа, информация о ходе строительства и др.) [11]. Таким образом, в каждый момент времени один и тот же информационный поток может содержать различную информацию.

В данном случае агенты являются как потребителями, так и производителями информации, но при этом каждый тип агентов имеет доступ лишь к определённой её части, в то же время не все области информационного пространства могут быть подвержены изменениям со стороны агентов.

Как и в реальном мире, информационное пространство влияет на исход взаимодействий между различными агентами – участниками реализации проекта, поскольку может содержать как справочную информацию, на основе которой агенты принимают решения, так и срочные сообщения, к примеру, информацию о произошедшей аварии т.д.

Поскольку в некоторых случаях при формировании и восприятии информации речь идёт о субъективном взгляде на те или иные события, несвоевременности получения каких-либо сведений, а также о других влияниях человеческого фактора, было решено в некоторых случаях восприятия информации агентами искусственно внести некоторое её искажение (по аналогии с реальными процессами: агент неверно интерпретирует считанную информацию

или ошибается при её обработке). Вероятность возникновения искажений будет варьироваться [9]. Кроме того, наблюдается очевидная необходимость обеспечения мультидоступа к данным, в связи с этим ведется разработка соответствующих протоколов на основе классического подхода [12–15].

Очевидно, что для реализации концепции взаимодействия со средой агент должен в той или иной степени воспринимать состояние своего окружения и уметь совершать действия, которые влияют на состояние. Кроме того, агент также должен иметь цели или задачи, связанные с состоянием окружающей среды. Взаимодействие между активным агентом, принимающим решения, и его средой, в рамках которой агент стремится достичь цель, характеризуется как тем, что среда влияет на агента, так и тем, что действия агента могут влиять на будущее состояние окружающей среды

Правильный (наиболее выгодный) выбор решения, которое агент принимает, может в перспективе исследования потребовать «предусмотрительности», то есть планирования – принятия тех или иных решений на следующих этапах взаимодействия со средой. И наоборот, агент может использовать свой опыт для повышения своей производительности с течением времени. Знания, которые агент привносит в задачу с самого начала (либо из предыдущего опыта работы агента, либо вложенные в него путем проектирования) влияют на то, что будет представлять выгоду для агента. Взаимодействие с окружающей средой необходимо для корректировки поведения с целью использования конкретных особенностей задачи.

5. Формализация среды окружения при помощи марковских процессов принятия решений с конечным числом этапов

Рассмотрим процесс взаимодействия агентов и окружающей среды: агент выбирает действия, а среда реагирует на эти действия и представляет агенту новые ситуации. С этой позиции в рамках данного исследования разрабатывается дискретная недетерминированная динамическая среда [16]. Иными словами, предполагается, что: а) агенты воспринимают события с достаточно высокой, но ограниченной точностью (информация, воспринятая агентом, может отличаться от информации, сформированной средой), поскольку имеет место искусственное искажение информации; б) в ответ на преобразование среды агент формирует неограниченное число реакций – от инициирования взаимодействия с другими агентами (или же формирования изменений в другой области информационной среды) до отсутствия какой-либо реакции; в) в момент между восприятием изменения среды и реакцией агента на это изменение информационная среда может преобразоваться вновь. Проектирование среды, обладающей данными характерными чертами, обусловлено целью достижения наибольшего сходства модели с процессами, протекающими в реальном мире.

Было решено формализовать процесс проектирования модели информационного пространства и способов взаимодействия с ним агентов, используя идеи теории динамических систем, в частности, таких как оптимальное управление марковскими процессами принятия решений [17]. Для этого необходимо охватить наиболее важные аспекты реальной проблемы, с которой сталкивается агент, взаимодействующий во времени со своей средой для достижения цели.

Агент должен в некоторой степени ощущать состояние своего окружения и уметь предпринимать действия, которые влияют на состояние среды, посредством сенсоров и эффекторов. Сенсоры – это абстракция, описывающая своего рода «датчики» для восприятия окружающей среды, а «эффекторы» – исполнительные органы, которые изменяют среду (реагируют на различного рода события). Важной задачей также является определение способов восприятия агентами изменений информационной среды (и обратно).

Агент также должен иметь цели или задачи, связанные с состоянием окружающей среды. Марковские процессы принятия решений призваны включать только эти три аспекта – ощущение, действие и цель – в их простейших возможных формах, но не упрощая излишне ни один из них. Цель агента в данном случае – специальный сигнал, называемый

вознаграждением, дающимся средой агенту. На каждом временном шаге награда – это число $R_t \in R$. Основная цель агента – максимизировать общую сумму вознаграждения, которое он получает с течением времени посредством выбора действий (рис. 1).

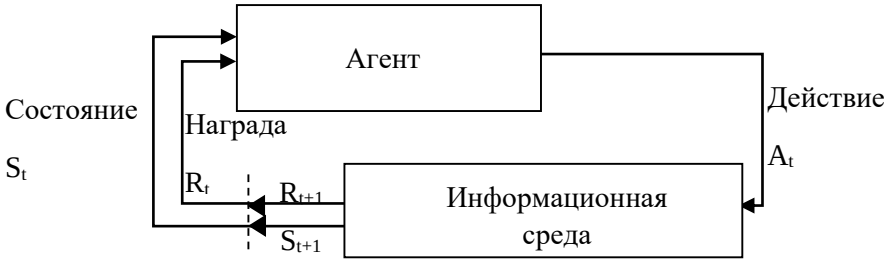


Рис. 1. Взаимодействие агента и среды в марковском процессе принятия решений

Введём формально проблему конечных марковских процессов принятия решений применительно к поставленной задаче. Эта проблема включает в себя ассоциативный аспект – выбор различных действий в различных ситуациях. Классическая формализация последовательного принятия решений соответствует тем случаям, когда действия влияют не только на немедленные награды, но и на последующие ситуации. Таким образом, марковские процессы принятия решений учитывают отложенное (задержанное) вознаграждение и необходимость компромисса немедленного и отложенного вознаграждения.

Агент и среда взаимодействуют на каждом из последовательных шагов дискретного времени $t = 0, 1, 2, 3, \ldots$ На каждом временном шаге t агент получает некоторое представление о состоянии среды $S_t \in S$ и на этом основании выбирает действие $A_t \in A(s)$, где $A(s)$ – набор действий. На данном этапе проектирования примем допущение, что $A(s)$ одинаков во всех состояниях. На следующем шаге времени в результате выбора своего действия агент получает числовое вознаграждение $R_{t+1} \in R \subset \mathbb{R}$ и попадает в новое состояние S_{t+1}.

Таким образом, можно описать процесс при помощи последовательности, которая начинается следующим образом:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

В конечном марковском процессе принятия решений наборы состояний, действий и вознаграждений (S, A и R) имеют конечное число элементов. В этом случае случайные величины R_t и S_t имеют четко определенные распределения вероятностей, зависящие только от предшествующего состояния и действия. То есть для конкретных значений этих случайных величин $s' \in S$ и $r \in R$ существует вероятность того, что эти значения будут заданы в момент времени t при определенных значениях предыдущего состояния и действия, которую можно вычислить при помощи функции p:

$$p(s', r \mid s, a) = Pr\{St = s', Rt = r \mid S_{t-1} = s, A_{t-1} = a\},\ \forall s', s \in S,\ r \in R \text{ и } a \in A(s).$$

Это определение функции p: $S \times R \times S \times A \to [0, 1]$, которая является обычной детерминированной функцией от четырех аргументов, p задает распределение вероятностей для каждого выбора s и a, то есть:

$$\sum_{s' \in S} \sum_{r \in R} p(s', r \mid s, a) = 1 \quad \forall s \in S, a \in A(s).$$

Вероятности, задаваемые функцией p, полностью характеризуют динамику конечного марковского процесса принятия решений. Исходя из этого, можно вычислить все, что можно знать об окружающей среде, например, вероятности перехода состояний (которые обозначим по определению как функцию p от трех элементов, p: $S \times S \times A \to [0, 1]$):

$$p(s' \mid s, a) = Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r \mid s, a).$$

Мы также можем вычислить ожидаемое вознаграждение для пар состояние – действие в виде функции r с двумя аргументами $r: S \times A \to \mathbb{R}$:

$$r(s,a) = E[R_t \mid S_{t-1} = s,\ A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r \mid s, a)$$

или же вычислить ожидаемое вознаграждение для троек состояние – действие – следующее состояние при помощи функции r от трёх аргументов:

$$r(s,a,s') = E[R_t \mid S_{t-1} = s,\ A_{t-1} = a,\ S_t = s'] = \sum_{r \in R} r \frac{p(s',\ r \mid s,\ a)}{p(s' \mid s,\ a)}.$$

Структура марковского процесса принятия решений является абстрактной, гибкой, а значит, может быть применена не только к поставленной задаче, но и ко многим другим проблемам различными способами. Общее правило, которого необходимо придерживаться, заключается в том, что все, что не может быть произвольно изменено агентом, считается находящимся вне его и, следовательно, является частью его среды. При этом агент может обладать некоторыми знаниями о том, как вычисляется его вознаграждение. Но осуществление процесса вычисления вознаграждения всегда должно быть внешним по отношению к агенту, поскольку это определяет задачу, стоящую перед агентом, и, следовательно, не может произвольно меняться. Фактически в некоторых случаях агент может знать все о том, как работает его среда, если это необходимо.

6. Заключение

На данном этапе исследования предложен прототип мультиагентной системы для моделирования процессов, сопряженных с реализацией транспортных мегапроектов. Особое внимание уделено описанию информационной среды функционирования агентов как ключевому элементу мультиагентной системы – это модель окружающей среды, которая имитирует формирование и изменение информационного пространства. Определён состав способов формирования и изменения её компонентов агентами и обратно, методов воздействия среды на агентов на основе теории конечного марковского процесса принятия решений.

В перспективе стоит задача уточнения агентной модели: состава агентов, алгоритмов их взаимодействия, развития топологии МАС, а также усовершенствования прототипа разрабатываемой информационной среды. Кроме того, представляется необходимой разработка коммуникационных протоколов для организации корректного взаимодействия агентов и их общего информационного пространства.

Литература

1. *Лычкина Н. Н.* Информационные системы управления производственной компанией: учебник и практикум для академического бакалавриата. М.: Изд. Юрайт, 2018.
2. *Жмурко С. А.* Обобщенная модель агента и многоагентной системы // Известия ЮФУ. Технические науки. 2008. № 4.
3. *Городецкий В. И., Грушинский М. С., Хабалов А. В.* Многоагентные системы // Новости искусств. интеллекта. 1998. № 2. С. 64–116.
4. *Хорошевский В. Ф.* Методы и средства проектирования и реализации мультиагентных систем // Матер. семинара «Проблемы искусственного интеллекта», ИПУ РАН, 1999.
5. *Маслобоев А. В.* Механизмы взаимодействия и координации агентов в открытой мультиагентной системе информационной поддержки региональных инновационных структур // Теория и практика системной динамики: труды II-ой Всерос. науч. конф., Апатиты, КНЦ РАН, 2007.

6. *Маслобоев А. В.* Мультиагентная технология информационной поддержки инновационной деятельности в регионе. Прикладные проблемы управления макросистемами // Труды Института системного анализа РАН. 2009. Т. 39.
7. *Замятина Е. Б.* Современные теории имитационного моделирования: специальный курс. Пермь: ПГУ, 2007.
8. *Аристов С. А.* Имитационное моделирование экономических процессов: учебное пособие. Екатеринбург, 2003.
9. *Пантелеев М. Г., Кохтенко Н. В., Лебедев С. В.* Среда имитационного моделирования агентных систем реального времени // Научно-технический вестник информационных технологий, механики и оптики. 2012. № 1 (77).
10. *Чайковский Д. В.* Информационное пространство: анализ определений // Вестник БГУ. 2010. № 14.
11. *Манойло А. В.* Государственная информационная политика в особых условиях: монография. М.: МИФИ, 2003.
12. *Дейт К. Дж*. Введение в системы баз данных. 7-е издание. М.: Изд. «Вильямс», 2001.
13. *Гарсиа-Молина Г., Ульман Д. Д., Уидом Д.* Системы баз данных. Полный курс. М.: Изд. «Вильямс», 2003.
14. *Малыхина М. П.* Базы данных: основы, проектирование, использование. СПб.: БХВ-Петербург, 2004.
15. *Диго С. М.* Базы данных: проектирование и использование: учебное пособие для вузов. М.: Финансы и статистика, 2005.
16. *Девятков В. В.* Системы искусственного интеллекта: учебное пособие для вузов. М.: Изд-во МГТУ им. Н. Э. Баумана, 2001.
17. *Richard S. Sutton, Andrew G. Barto* Reinforcement Learning: An Introduction. The MIT Press Cambridge, Massachusetts.

Статья поступила в редакцию 31.07.2019;
переработанный вариант – 02.09.2019.

Есикова Татьяна Николаевна
к.э.н., ведущий научный сотрудник ИЭиОПП СО РАН (630090, Новосибирск, пр. Академика Лаврентьева, 17), тел. (383) 330-25-96, e-mail: T.N.Yesikova@gmail.com.

Вахрушева Светлана Витальевна
аспирант НГУ (630090, Новосибирск, ул. Пирогова, 1), e-mail: s.vakhr@gmail.com.

Agent environment simulation while developing multi-agent system with large-scale infrastructure projects serving as an example

T. Yesikova, S. Vakhrusheva

The article poses the problem of simulation processes associated with the construction of large-scale infrastructural projects in the context of the Bering Strait tunnel. The research is based on such a simulation method as a multi-agent modeling. The article describes the basics of building a multi-agent system environment in relation to this task. It contains description of the environment interactions and identified agents.

Keywords: large-scale infrastructural projects, simulation modelling, multi-agent system (MAS), agents, interaction environment, MAS architecture.

УДК 004.827:004.89:519.816:620.9

Разработка интеллектуальной СППР по предотвращению угроз энергетической безопасности

Г. Б. Загорулько, Л. В. Массель[1]

В статье описывается интеллектуальная система поддержки принятия решений (ИСППР) по предотвращению угроз энергетической безопасности. При ее создании была использована методика, базирующаяся на современных подходах и принципах разработки систем такого класса. ИСППР предоставляет доступ к систематизированной информации об угрозах энергетической безопасности и помогает выбрать превентивные мероприятия при решении двух конкретных задач, моделирующих угрозу похолодания и аварию у производителя энергоресурса.

Ключевые слова: энергетическая безопасность, интеллектуальные системы поддержки принятия решений, методика разработки интеллектуальных СППР, ИСППР по предотвращению угроз энергетической безопасности.

1. Введение

Понятие энергетической безопасности (ЭнБ) является одной из основных составляющих Концепции национальной безопасности Российской Федерации. Вопросами ЭнБ занимаются многие научные коллективы и государственные структуры. Правительством РФ принята Энергетическая стратегия России на период до 2030 г. Утверждена разработанная в Минэнерго новая Доктрина энергетической безопасности Российской Федерации [1].

Существует много определений ЭнБ, отражающих разные аспекты этого понятия. В институте систем энергетики им. М. А. Мелентьева СО РАН (ИСЭМ СО РАН) в 1996 г. было предложено и позднее рекомендовано для использования специалистами отраслей энергетики следующее определение: «энергетическая безопасность – это состояние защищенности граждан, общества, государства, экономики от угроз дефицита в обеспечении их потребностей в энергии экономически доступными энергетическими ресурсами приемлемого качества, от угроз нарушений бесперебойности энергоснабжения. При этом состояние защищенности – состояние, соответствующее в нормальных условиях обеспечению в полном объеме обоснованных потребностей (спроса) в энергии, в экстремальных условиях – гарантированному обеспечению минимально необходимого объема потребностей» [2].

ИСЭМ СО РАН на протяжении ряда лет занимается исследованиями проблем ЭнБ. Определение и моделирование угроз ЭнБ, разработка превентивных мероприятий являются важными актуальными задачами, решение которых осуществляется в рамках этих исследований. В частности, разработан ряд семантических моделей и прототипов основных компонентов СППР в исследованиях и обеспечении ЭнБ [3], выполнен онтологический инжиниринг предметной области «Энергетическая безопасность» [4].

[1] Работа выполнена при частичной финансовой поддержке РФФИ (гранты № 19-07-00762, № 19-07-00351).

Описанная в статье интеллектуальная СППР (ИСППР) является совместной разработкой коллективов ИСЭМ СО РАН и Института систем информатики им. А. П. Ершова (ИСИ СО РАН). Для ее создания была использована методика, предложенная в ИСИ СО РАН, которая объединяет современные подходы и принципы создания ИСППР [5] и разработанные в ИСЭМ СО РАН онтологии и семантические модели [3, 4]. Рассматриваемая ИСППР предоставляет доступ к систематизированной информации о предметной области «Угрозы энергетической безопасности» и помогает выбрать превентивные мероприятия при решении двух конкретных задач: моделирование угрозы похолодания [6, 7] и моделирование аварии у производителя энергоресурса [8].

2. Существующие подходы и принципы разработки ИСППР

Разработка ИСППР является сложной важной задачей. Для ее решения предложено много подходов как в России, так и за рубежом.

В работе [9] дан ретроспективный обзор подходов к разработке СППР. В работе [10] подчеркивается, что не существует наилучшего стандартного подхода или методологии для разработки СППР. Автор выделяет три основных подхода, указывая их плюсы и минусы: 1) подход, обеспечивающий полный жизненный цикл системы (SDLC); 2) быстрое прототипирование и 3) подход конечного пользователя, при котором ЛПР может сам выбирать средства для решения своих задач. Второй и третий подходы позволяют быстро получить рабочую систему. Наиболее распространенным, универсальным и практически значимым автор считает подход быстрого прототипирования, который позволяет быстро оценить возможности полученной системы и при необходимости выполнить ее модификацию. Такой подход способствует объединению усилий лица, принимающего решения (ЛПР), и аналитика.

В последнее время много говорится о целесообразности использования методологии гибкой разработки программного обеспечения (Agile software development) при создании СППР [11]. Такие принципы этой методологии, как использование итеративной разработки, динамическое формирование требований, тесное взаимодействие разработчиков различного профиля, перекликаются с принципами быстрого прототипирования.

О подходах к созданию медицинских СППР упоминается в [12]. Авторы считают перспективными как разработку специализированных систем, так и использование гибких, легких сервисов, позиционируемых как отдельные приложения и интегрируемых между собой.

Об унифицированном подходе к разработке программного обеспечения СППР говорится в работе [13]. Данный подход подразумевает использование каркасной архитектуры программного обеспечения и следующих принципов проектирования: принцип свободы от субъективизма разработчиков, позволяющий системе быть максимально гибкой и подстраиваться под нужды и потребности ЛПР; принцип инвариантности по отношению к предметной области; принцип множественности методов, при котором СППР обеспечивает легкую и прозрачную интеграцию требуемых методов, советует пользователю, какой метод лучше применим и почему.

Популярным является подход, когда СППР строится для решения определенного класса задач, содержит множество методов и позволяет выбирать методы для решения конкретных задач. Описываемая в работе [14] экспертная СППР (ЭСППР) содержит около 50 методов принятия решений, а также базу знаний, представленную в виде последовательности вопросов об элементах задачи принятия решения и их различных реализациях (ответах). ЭСППР является web-приложением, что делает ее доступной широкому кругу пользователей и позволяет привлекать экспертов для оценки альтернативных вариантов решения. Данная ЭСППР ориентирована на решение экономических задач.

В настоящее время общепризнанным является мнение, что современная ИСППР должна быть информационной системой с развитыми средствами анализа и обработки хранящихся в ней данных [15]. СППР, предоставляющие доступ через интернет, описаны в работах [16, 17].

3. Используемая методика разработки ИСППР

В данной работе использовалась методика разработки ИСППР, предложенная в [5], которая объединяет современные подходы и принципы создания систем такого класса. На рис. 1 показаны её основные элементы. Разработка ИСППР основывается на таких принципах, как Максимальное использование готовых решений, Информативность, Доступность, Модульность, Открытость, Масштабируемость, Независимость от предметной области. При разработке используются Онтологический, Фрактально-стратифицированный [18], Сервис-ориентированный подходы, подходы Быстрого прототипирования и Гибкой разработки, Технологии Semantic Web, и Технология разработки информационно-аналитических интернет-ресурсов (ИАИР) [19].

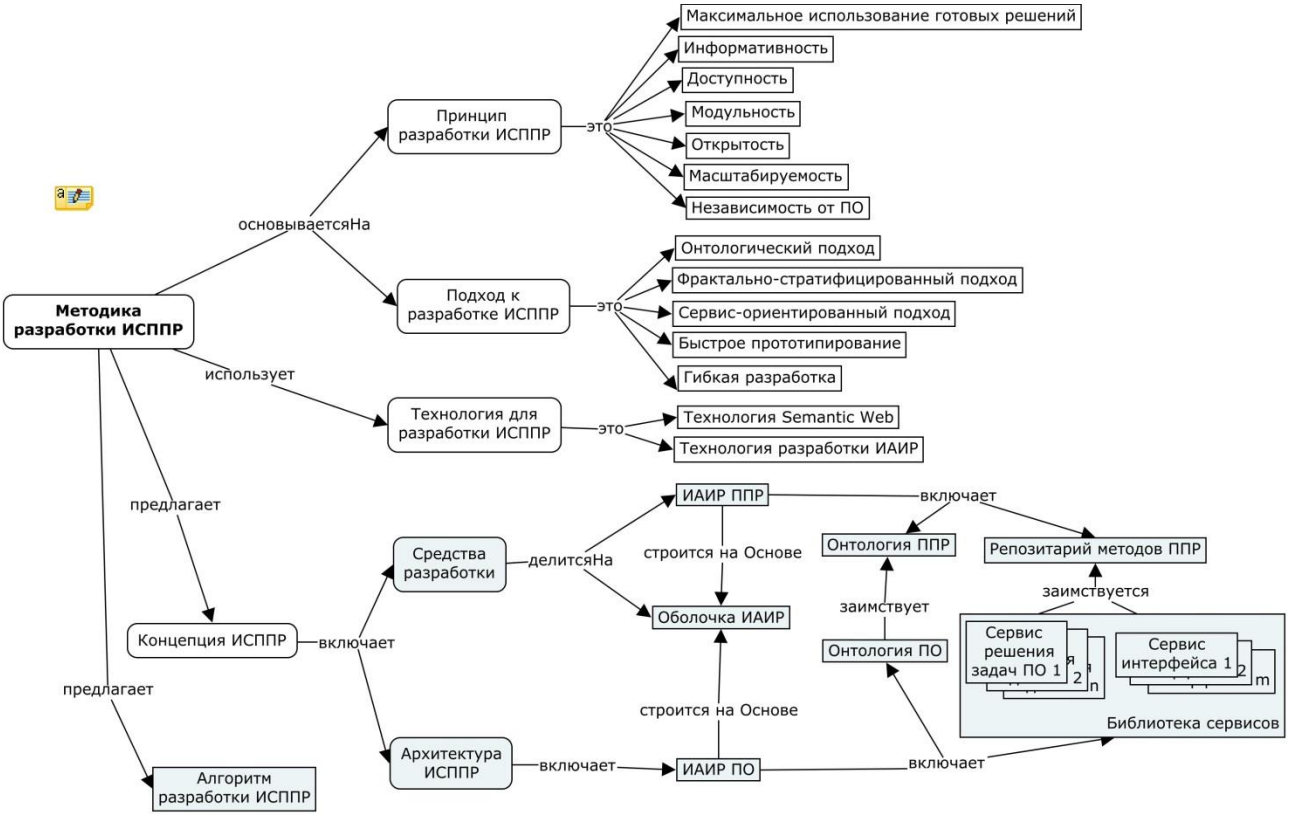


Рис. 1. Методика разработки ИСППР

Предлагаемые рассматриваемой методикой средства позволяют получать быстрые прототипы СППР, легко осуществлять их модификацию за счет использования онтологии области знаний (ОЗ) «Поддержка принятия решений» (ППР), построенного на её основе ИАИР, репозитария методов ППР и оболочки ИАИР [5]. Онтология ППР предоставляет концептуальный базис и позволяет организовать взаимодействие всех типов разработчиков ИСППР (экспертов, инженеров знаний, программистов). Фрактально-стратифицированный подход к разработке онтологии позволяет представить ОЗ ППР как систему подобных друг другу страт, в которых понятия описаны с точки зрения разных типов разработчиков [20]. ИАИР ППР предоставляет доступ к систематизированной на основе онтологии информации об ОЗ ППР. Методы ППР, входящие в репозитарий, реализованы в виде сервисов, что обеспечивает масштабируемость репозитария и позволяет легко интегрировать методы между собой. ИАИР ППР является каркасом разрабатываемой ИСППР, к которому подключаются необходимые сервисы, обеспечивающие её функциональность.

Предлагаемые средства для разработки ИСППР не зависят от их предметных областей, однако для облегчения представления знаний об этих областях разработчики могут восполь-

зоваться набором базовых онтологий и паттернов онтологического проектирования [21], методикой построения онтологий. Ресурс, созданный на основе этой онтологии и оболочки ИАИР, является информационной системой, систематизирующей знания о предметной области ИСППР и предоставляющей к ним содержательный доступ. Подключенные к ресурсу сервисы из репозитария делают его полноценной ИСППР и позволяют решать поставленные перед ней задачи.

В рамках предложенной методики были разработаны архитектура (рис. 2) и алгоритм построения типовой СППР, которые были конкретизированы для нескольких областей, в т.ч. для энергетики.

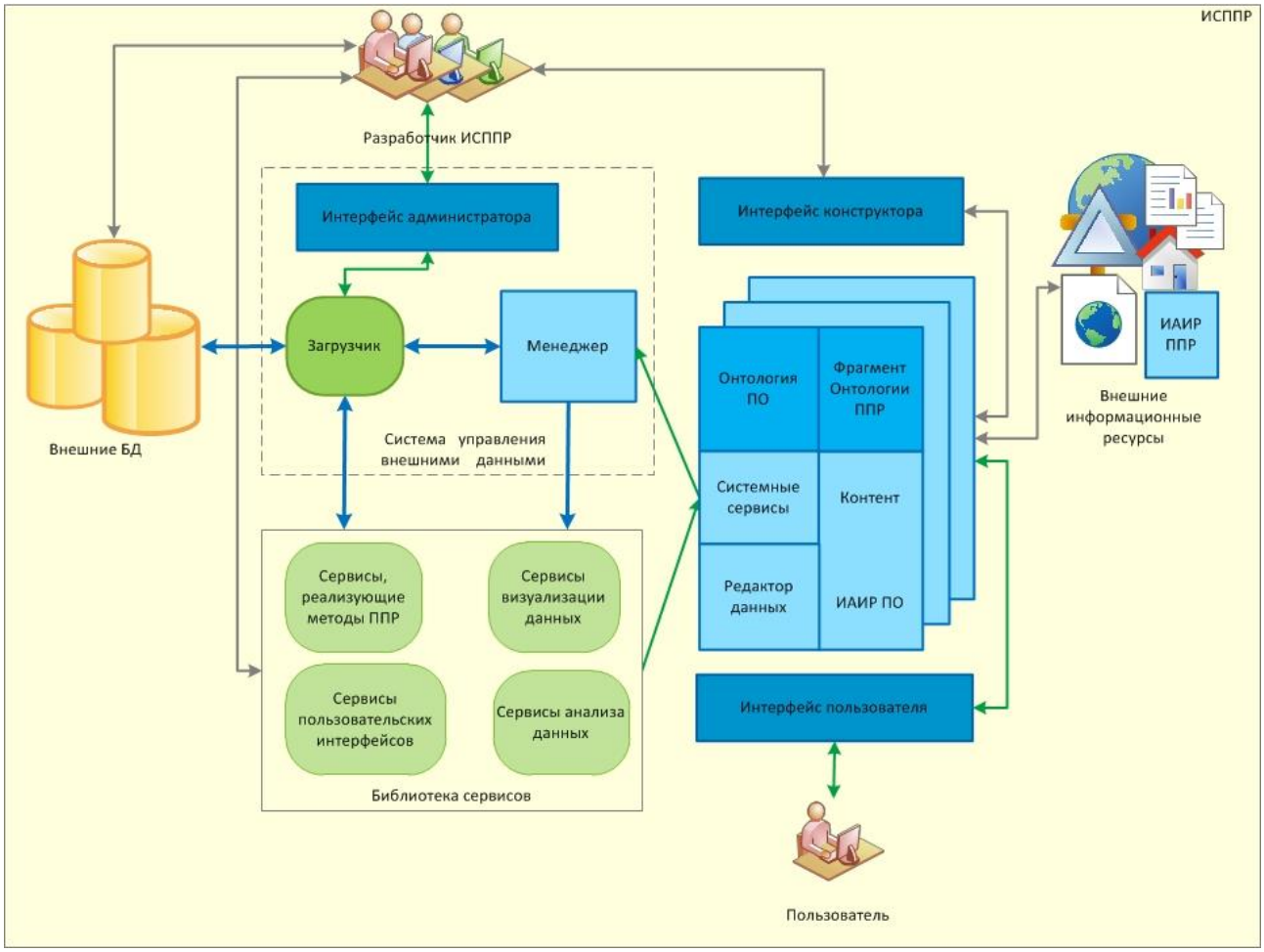


Рис. 2. Архитектура типовой ИСППР

4. Разработка ИСППР «Угрозы энергетической безопасности»

Рассматриваемая ИСППР построена на основе системного анализа предметной области «Угрозы энергетической безопасности». Были выделены цели, задачи, объекты, предметы, субъекты, методы исследования [2, 6, 8], использована онтология угроз ЭнБ, фрагмент которой представлен на рис. 3.

Предполагалось, что ИСППР «Угрозы энергетической безопасности» будет ориентирована на выбор превентивных мероприятий для решения конкретных задач в области энергетической безопасности. Были построены модели двух задач, с помощью ресурса ИАИР ППР выбраны методы и средства их решения. В онтологии ППР выделен фрагмент, описывающий классы, к которым принадлежат решаемые ИСППР конкретные задачи и их связи с другими

классами онтологии. Выделенный фрагмент был встроен в Онтологию угроз ЭнБ. В репозитарии методов ППР выделены методы, решающие задачи рассматриваемой ИСППР, и встроены в Библиотеку сервисов (рис.1).

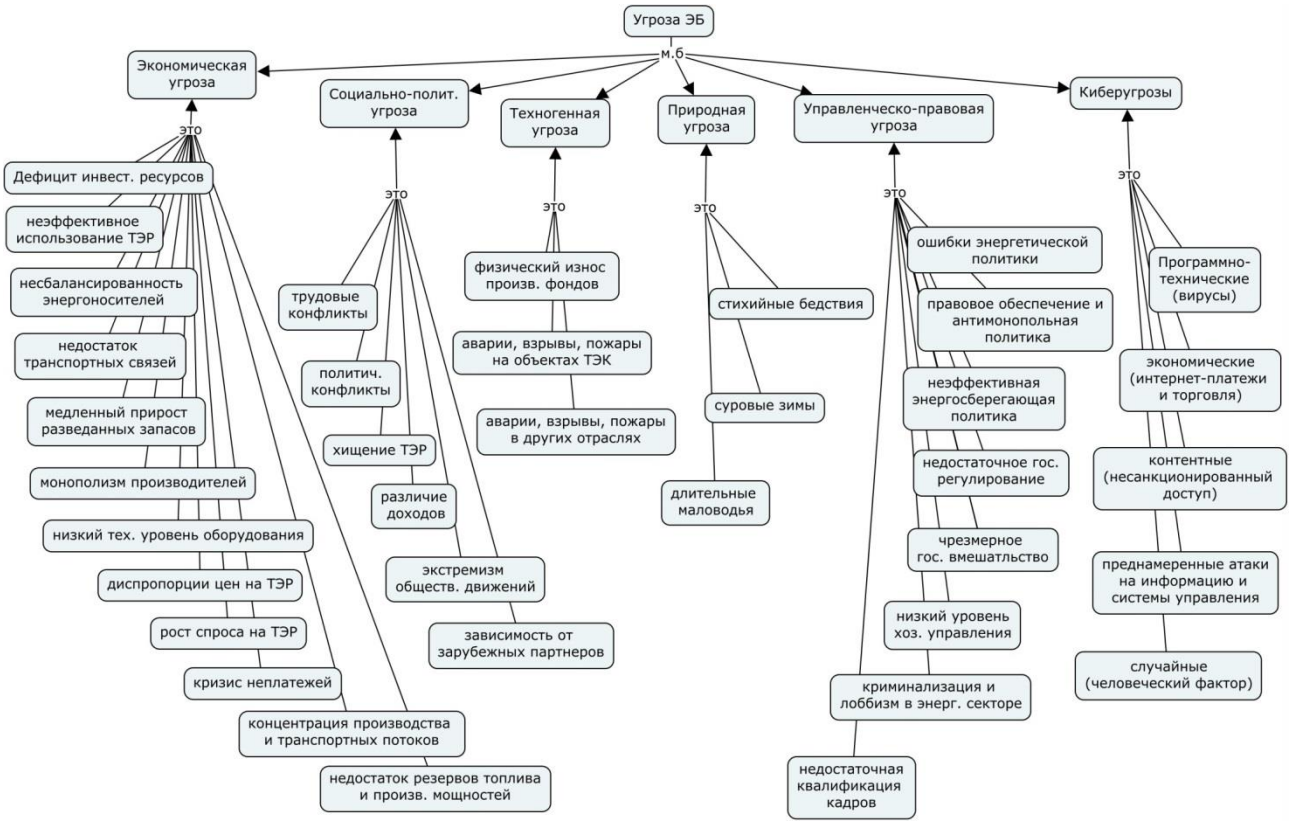


Рис. 3. Иерархия угроз ЭнБ

На основе Онтологии угроз ЭнБ был разработан ресурс по угрозам ЭнБ (рис. 4). В его контент добавлены информационные объекты, описывающие решаемые ИСППР задачи. Каждый такой информационный объект содержит неформальное описание задачи, связи с другими объектами онтологии – предметами и объектами исследования разрабатываемой ИСППР, угрозами, для предотвращения которых она предназначена, публикациями, в которых описывается задача, методами ППР и программными разработками, подходящими для решения данной задачи, необходимыми входными данными и способами их представления, сервисами, позволяющими задать входные данные, запустить процесс решения задачи и посмотреть результат.

ИСППР «Угрозы энергетической безопасности» решает две конкретные задачи, позволяющие определить превентивные мероприятия для обеспечения бесперебойного энергоснабжения потребителей в случаях реализации угрозы похолодания [6, 7] и угрозы аварии у производителя энергоресурса [8].

В первой задаче моделируется ситуация, когда из-за снижения температуры наружного воздуха относительно среднемноголетней в определенные месяцы отопительного периода повышается потребление тепло- и электроэнергии. Модель ситуации отражает аналитические зависимости между этими параметрами и позволяет выбрать превентивные меры (оценить, насколько нужно увеличить выработку указанных видов энергии, чтобы не допустить ее дефицита).

Во второй задаче моделируется реакция сети энергоснабжения на угрозы технологического типа. В случае аварийной ситуации у какого-либо производителя подключаются пункты резервного хранения энергоресурсов. Суммарное содержание ресурса в таких пунктах должно компенсировать его сокращение у производителей в результате аварии. Любой потребитель может получать ресурс из любого пункта резервного хранения, т.е. потенциально между ними

можно установить связь. В модели задачи задаются веса таких связей. Цель решаемой задачи – оптимизировать потоки ресурсов из резервных хранилищ потребителям с учетом весов их связей.

Рис. 4. ИСППР «Угрозы энергетической безопасности»

Для решения этих задач был выбран метод недоопределенных вычислений [22]. Данный метод позволяет представить модель решаемой задачи в виде набора параметров, связанных математическими соотношениями, без разделения параметров на входные и выходные. Параметры могут задаваться интервальными значениями, которые сужаются в процессе вычислений. Для каждой задачи были реализованы сервисы, позволяющие пользователям ИСППР задавать значения параметров модели, запускать модели на счет и отображать полученные результаты (рис. 5).

Рассмотрим более подробно использование ИСППР «Угрозы энергетической безопасности» и метода недоопределенных вычислений для выбора превентивных мероприятий при угрозе похолодания. Модель этой ситуации представляется следующими соотношениями:

$$\Delta B_{\text{т/э}} = \frac{\Delta t_{\text{нар}}}{t_{\text{вн}} - t_{\text{нар.ср.м.}}} \times 100\ \%,$$

где $\Delta B_{\text{т/э}}$ – изменение потребления теплоэнергии вследствие похолодания, $\Delta t_{\text{нар}}$ – отклонение температуры наружного воздуха от средней в рассматриваемом месяце, $t_{\text{вн}}$ – температура воздуха отапливаемых помещений, $t_{\text{нар.ср.м.}}$ – средняя температура наружного воздуха в рассматриваемом месяце.

$$\Delta B_{\text{э/э}} = 0.3 \times \Delta B_{\text{т/э}},$$

где $\Delta B_{\text{э/э}}$ – изменение потребления электроэнергии, принимаемое равным 30 % от изменения потребления тепла $\Delta B_{\text{т/э}}$:

$$B_{\text{т/э}}^{\text{дефицит}} = \frac{B_{\text{т/э}}^{\text{норм}} * \Delta B_{\text{т/э}}}{100} - D_{\text{т/э}}^{\text{кот}} - D_{\text{т/э}}^{\text{ТЭЦ}},$$

где $B_{\text{т/э}}^{\text{дефицит}}$ – дефицит теплоэнергии вследствие похолодания (Гкал); $B_{\text{т/э}}^{\text{норм}}$ – нормальное потребление теплоэнергии (Гкал); $\Delta B_{\text{т/э}}$ – увеличение потребления теплоэнергии (%); $D_{\text{т/э}}^{\text{кот}}$ – увеличение выработки тепла на котельных (Гкал), $D_{\text{т/э}}^{\text{ТЭЦ}}$ – увеличение выработки тепла на ТЭЦ (Гкал).

$$B_{\text{э/э}}^{\text{дефицит}} = \frac{B_{\text{э/э}}^{\text{норм}} * \Delta B_{\text{э/э}}}{100} - D_{\text{э/э}}^{\text{ТЭС}},$$

где $B_{\text{э/э}}^{\text{дефицит}}$ – дефицит электроэнергии вследствие похолодания (кВт*ч); $B_{\text{э/э}}^{\text{норм}}$ – нормальное потребление электроэнергии (кВт*ч); $\Delta B_{\text{э/э}}$ – увеличение потребления электроэнергии (%); $D_{\text{э/э}}^{\text{ТЭС}}$ – увеличение выработки электроэнергии на ТЭС (кВт*ч).

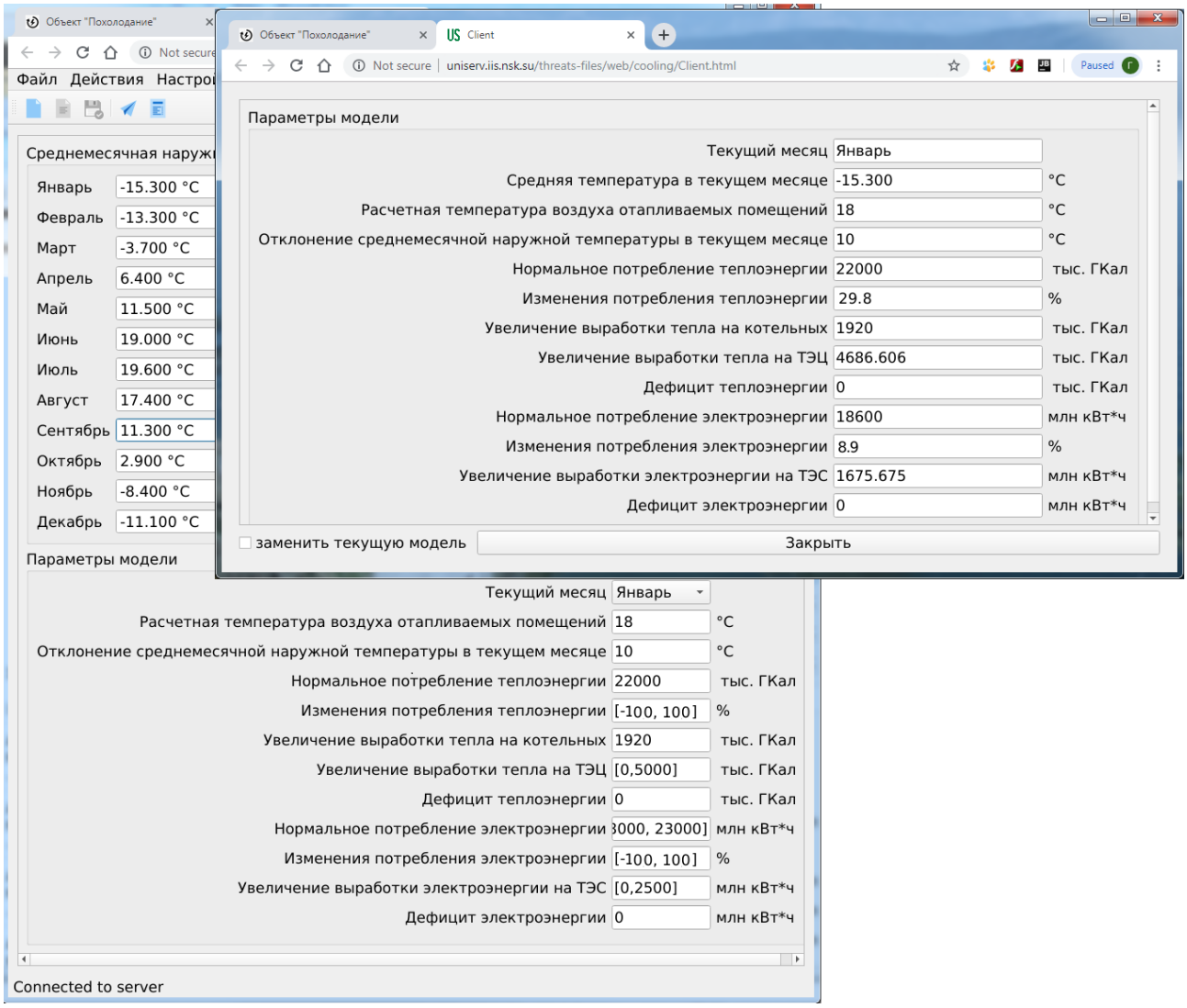


Рис. 5. Исходные данные и результаты решения задачи, моделирующей угрозу похолодания

Значения среднемесячных температур и нормального потребления энергоресурсов в определенных регионах и в определенные периоды берутся из источников статистических данных. Метод недоопределенных вычислений позволяет задавать значения нормального потребления тепло- и электроэнергии за определенный месяц интервалами, охватывающими значения этих параметров за несколько лет. Запустив процесс вычислений, пользователь в общем случае получает интервальные значения остальных параметров. Задавая более узкие значения одних параметров в пределах полученных интервалов, он получает суженные, а возможно, и точные значения других.

На рис. 5 показаны элементы пользовательского интерфейса с исходными данными и результатами решения рассматриваемой задачи. Поскольку метод не разделяет параметры на входные и выходные, на одной и той же модели можно решать как прямые, так и обратные задачи. Так, задав значения изменения температуры, нормального потребления энергоресурсов и значения увеличения их выработки, получим значения дефицита тепло и электроэнергии. А задав нулевые значения дефицита, узнаем, насколько нужно увеличить выработку энергоресурсов.

5. Заключение

Предлагаемая методика разработки ИСППР основана на современных подходах и принципах построения систем данного класса. Предлагаемые ею средства позволяют быстро строить ИСППР в различных слабоформализованных областях. Методика была использована для разработки ИСППР «Угрозы энергетической безопасности». Каркасом данной ИСППР является ИАИР, построенный на основе онтологии угроз ЭнБ. ИСППР предоставляет доступ к систематизированной информации об угрозах ЭнБ и решает две конкретные задачи: моделирование угрозы похолодания и моделирование аварии у производителя энергоресурса. Дальнейшее развитие методики предполагает расширение репозитария методов ППР, в частности разработку сервисов взаимодействия с конечными пользователями.

Литература

1. Доктрина ЭБР [Электронный ресурс]. URL: `https://minenergo.gov.ru/node/14766` (дата обращения: 16.07.2019).
2. Энергетическая безопасность России: проблемы и пути решения / Н. И. Пяткова [и др.]; отв. ред. Воропай Н. И.; Рос. акад. наук, Сиб. отд-ние, Ин-т систем энергетики им. Л. А. Мелентьева. Новосибирск: Изд-во СО РАН, 2011. 198 с.
3. *Массель Л. В., Массель А. Г.* Технологии и инструментальные средства интеллектуальной поддержки принятия решений в экстремальных ситуациях в энергетике // Вычислительные технологии. 2013. Т. 18. Специальный выпуск. С. 37–44.
4. *Массель Л. В., Ворожцова Т. Н., Пяткова Н. И.* Онтологический инжиниринг для поддержки принятия стратегических решений в энергетике // Онтология проектирования. 2017. Т. 7, № 1 (23). С. 66–76. DOI: 10.18287/2223-9537-2017-7-1-66-76.
5. *Загорулько Г. Б.* Методические аспекты разработки интеллектуальных СППР в слабоформализованных предметных областях на основе информационно-аналитических ресурсов // Труды 16-й Национальной конференции по искусственному интеллекту с международным участием (КИИ-2018), 24–27 сентября 2018 г., Москва. Т. 1. С. 36–43.
6. *Пяткова Н. И., Массель Л. В., Массель А. Г.* Методы ситуационного управления в исследованиях проблем энергетической безопасности // Известия Академии наук. Энергетика. 2016. № 4. С. 156–163.

7. *Массель А. Г., Пяткова Е. В.* Интеллектуальные информационные технологии для исследований проблем энергетической безопасности // Труды Всероссийского семинара с международным участием «Методические вопросы исследования надежности больших систем энергетики»: Вып. 64. Надежность систем энергетики: достижения, проблемы, перспективы. Иркутск: ИСЭМ СО РАН. 2014. С. 472-483. ISBN 978-5-93908-115-3.
8. *Карпов В. В., Симанчев Р. Ю.* Определение и угрозы энергетической безопасности // Вестник Омского университета. Серия «Экономика». 2016. № 4. С. 30–38.
9. *Garaibeh N. K., Al-Raddadi A.* Survey of DSS Development Methodologies. Encyclopedia of Business Analytics and Optimization, 2014. https://www.researchgate.net/publication/267625331_Survey_of_DSS_Development_Methodologies (date views: 06.02.2019).
10. *Power D. J.* Decision Support Systems Hyperbook. Cedar Falls, IA:DSSResources.COM, 2000. http://dssresources.com/subscriber/password/dssbook (date views: 06.02.2019).
11. *Garaibeh N. K.* DSS Development and Agile Methods: Towards a newFramework for Software Development Methodology // International Journal of Machine Learning and Computing. 2012. V. 2, № 4. https:// pdfs.semanticscholar.org/d2be/d1bd3a1eb792a774145826177f2cf5dbb803.pdf (date views: 06.02.2019).
12. *Ефименко И. В., Хорошевский В. Ф.* Интеллектуальные системы поддержки принятия решений в медицине: ретроспективный обзор состояния исследований и разработок и перспективы // Материалы конференции ОСТИС, Минск, Белоруссия, февраль 2017. № 7. С. 251–260
13. *Судаков В. А., Осипов В. П.* Унификация разработки программного обеспечения прикладных СППР // Материалы Всероссийского совещания по проблемам управления, Москва, 2014. С. 8855–8863
14. *Кравченко Т. К.* Развитие экспертной системы поддержки принятия решений // Искусственный интеллект и принятие решений. 2013. № 4. С. 72–80.
15. *Халин В. Г., Чернова Г. В., Юрков А. В.* Методологические аспекты создания и функционирования систем поддержки принятия решений // Экономический анализ: теория и практика. 2015. № 7 (406). С. 20–33.
16. *Edwards M. P., Lippeveld T., Khatri U., Kunaka D., Mwebaze M., Tohouri R.* Building a Web-Based Decision Support System. MEASURE Evaluation. 2018. https://www.measureevaluation.org/resources/publications/fs-18-288 (date views: 06.02.2019).
17. *Zhang D., Chen X., Yao H.* Development of a Prototype Web-Based Decision Support System for Watershed Management // Water. 2015. № 7. P. 780–793.
18. *Массель Л. В.* Фрактальный подход к структурированию знаний и примеры его применения // Онтология проектирования. 2016. Т. 6, № 2 (20). С. 149–161.
19. *Загорулько Ю. А., Загорулько Г. Б., Боровикова О. И.* Технология создания тематических интеллектуальных научных интернет-ресурсов, базирующаяся на онтологии // Программная инженерия. 2016. Т. 7, № 2. С. 51–60.
20. *Загорулько Г. Б.* Разработка онтологии для интернет-ресурса поддержки принятия решений в слабоформализованных областях // Онтология проектирования. 2016. Т. 6, № 4 (22). С. 485–500.
21. *Zagorulko Y., Zagorulko G., Borovikova O.* Implementation of Content Patterns in the Methodology of the Development of Ontologies for Scientific Subject Domains // Communications in Computer and Information Science. 2018. V. 934. P. 260–272.
22. *Нариньяни А. С., Телерман В. В., Ушаков Д. М., Швецов И. Е.* Программирование в ограничениях и недоопределённые модели // Информационные технологии. 1998. № 7. С. 13–22.

Статья поступила в редакцию 23.07.2019;
переработанный вариант – 30.08.2019.

Загорулько Галина Борисовна
н.с, Институт систем информатики имени А. П. Ершова СО РАН (630090, Новосибирск, пр. Ак. Лаврентьева, 6), e-mail: gal@iis.nsk.su.

Массель Людмила Васильевна
д.т.н., профессор, зав. лабораторией информационных технологий в энергетике, Институт систем энергетики им. Л. А. Мелентьева СО РАН (664033, Иркутск, ул. Лермонтова, 130), e-mail: massel@isem.irk.ru.

Development of intelligent DSS for preventing energy security threats

G. Zagorulko, L. Massel

The paper considers an intelligent decision support system (IDSS) for preventing energy security threats. This system is created by means of a methodology that combines modern approaches and principles for developing systems of this class. IDSS provides access to systematized information on energy security threats and helps to choose preventive measures for solving two specific tasks: simulation of the cooling threat and accident simulation caused by energy producer.

Keywords: energy security, intelligent decision support systems, the methodology for the development of intelligent DSS, IDSS for energy security threats.

УДК 004.912

Комплексный подход к исследованию лексических характеристик текста

Е. А. Сидорова[1]

В работе предлагается подход и рассматривается программное обеспечение для многоцелевого исследования лексических характеристик текста. Данная работа лежит на стыке корпусной лингвистики и лексикографических исследований. Основой проводимых исследований является корпус текста и создаваемый на его основе проблемно-ориентированный словарь. Необходимое программное обеспечение поддержки исследователя включает интерфейсы для разработки словарей, построения системы признаков, разметки терминов, а также средства автоматической генерации лексического наполнения словаря по текстам, поиска контекстов терминов, накопление статистической информации и др. При извлечении терминов осуществляется морфологический анализ текста и построение словосочетаний на основе правил согласования грамматических характеристик слов. Для исследования контекстов употребления терминов предоставляются средства построения конкордансов, что позволяет конечному пользователю пронаблюдать грамматические, семантические, стилистические и проблемно-ориентированные особенности терминов и осуществить их разметку.

Ключевые слова: термин, предметный словарь, корпус текстов, конкорданс.

1. Введение

Текст как источник и средство передачи информации нуждается во всестороннем исследовании, необходимом как для оценки качества изложенного, так и при автоматической обработке и поддержке языковых поисковых сервисов. Изучение языковых явлений и моделирование процессов понимания текста на разных языковых уровнях является фокусом современных исследований в компьютерной лингвистике. Для решения данных задач, как правило, используют разнообразные знания в формализованном виде, такие как тезаурусы (например, WordNet), толково-комбинаторные словари, аннотированные корпуса текстов (например, Национальный корпус русского языка, `www.ruscorpora.ru`) и т.п. Тезаурус как инструмент описания предметной лексики позволяет характеризовать термин и его связи с точки зрения особенностей употребления в данной предметной области [1]. Другим вариантом исследования языковых явлений является использование корпусов текстов. Корпус является источником и инструментом многоаспектных лексикографических работ [2], позволяя на основе разметки автоматизировать создание и начальное наполнение словарей.

Анализ литературы [3–5] показывает, что при извлечении терминологии из большого массива текстов используются подходы, объединяющие лингвистические и статистические методы. Результатом такого рода исследований может стать аннотированный корпус текстов, который позволит в дальнейшем осуществлять частотный анализ лексики, создавать конкор-

[1] Исследование выполнено при поддержке РФФИ в рамках проектов №17-07-01600 и № 18-00-01376 (18-00-00889).

дансы по различным основаниям и строить электронные словари. Использование специализированных методов позволит автоматизировать работу экспертов по исследованию формальных структур и дискурса в целом и построению лингвистических моделей на основе размеченного корпуса текста.

Данная работа посвящена описанию методики и инструментов, обеспечивающих исследования лексических характеристик текста на основе корпусов текстов. Совокупность предложенных инструментов позволяет сформировать среду для создания проблемно-ориентированных словарей и предоставить конечному пользователю различные возможности исследования языковых явлений.

2. Среда поддержки исследования текста

Разработка моделей и создание качественных ресурсов требует кропотливого ручного труда, поддерживаемого программным инструментарием. Программная среда должна предоставлять специалистам различные рабочие инструменты для формирования необходимых баз знаний и проведения корпусных исследований.

В рамках данной работы были сформулированы функциональные требования, которым среда должна удовлетворять:

a) обеспечивать автоматическое наполнение словарей на базе корпусов текстов;

b) предоставлять возможность настраивать и приписывать различные характеристики терминам словаря;

c) осуществлять лексический анализ текста – сегментацию и извлечение из текста заданных в словаре терминов;

d) обеспечивать накопление данных о статистико-комбинаторных свойствах обнаруженных в тексте языковых явлений;

e) осуществлять построение конкордансов терминов и их визуализацию.

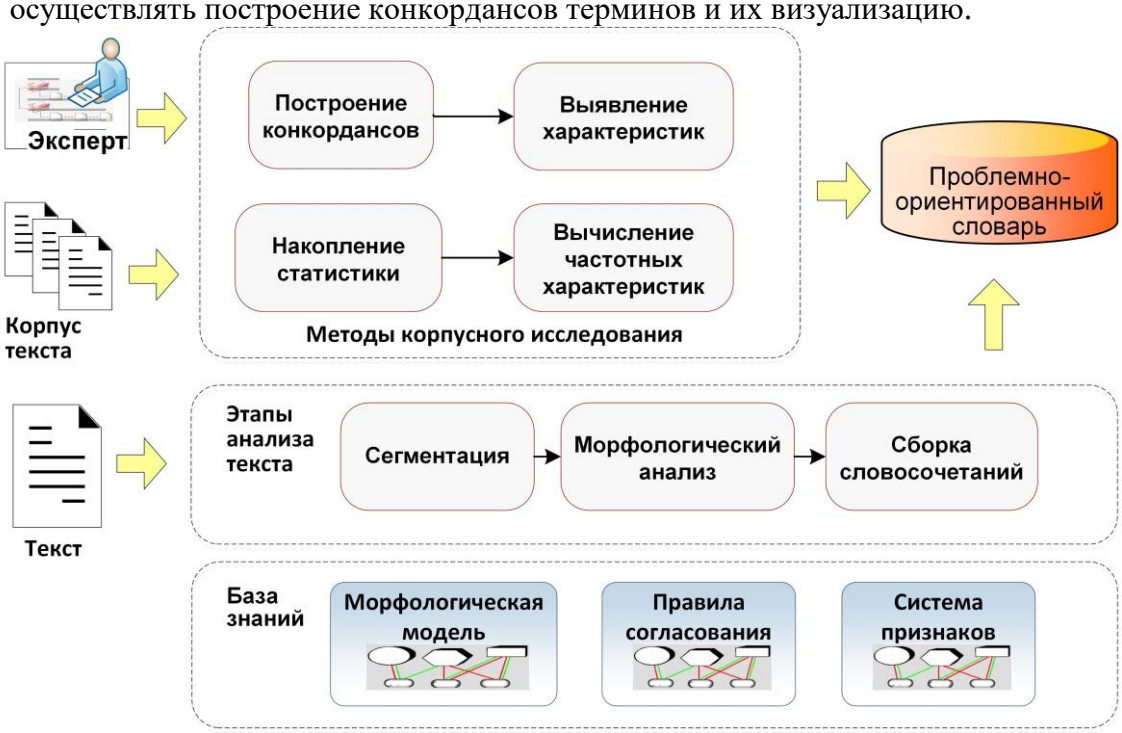


Рис. 1. Среда исследования лексических характеристик текста

Созданная система включает следующие базовые инструменты для проведения исследований (рис. 1): интерфейс разработки словаря и построения системы признаков, средства автоматической генерации лексического наполнения словаря по корпусу текстов и вычисления

количественных характеристик найденных терминов, инструменты построения конкордансов для исследования контекстов лексических единиц.

3. Модель представления знаний

Рассматриваемая лексикографическая модель знаний включает три основных компонента. Словарь задает лексическую модель рассматриваемого подъязыка, определяемого проблемной областью. Грамматика обеспечивает поиск и извлечение лексических единиц из текстов. Система прагматически ориентированных характеристик, задаваемая пользователем, поддерживает фиксацию наблюдений и ориентирована на дальнейшую поддержку автоматизированных методов обработки текста.

Базой исследований является представительный проблемно-ориентированный корпус текстов. Основным инструментом, обеспечивающим поддержку исследований, является поиск примеров употребления терминов словаря, построение множества контекстов (конкордансов), вычисление количественных характеристик встречаемости, совместной встречаемости, распределения и др.

3.1. Лексическая модель

В рамках нашего подхода в словарной статье хранится необходимая информация как для извлечения термина из текста, так и для поддержки последующих этапов анализа.

Проблемно-ориентированный словарь – это объем лексики, организованной по семантическому (тематическому / жанровому / и др.) принципу с отражением определенного набора базовых формальных отношений. Формально словарь описывается системой вида:

$D = \{W, P, M, G, S, F_w, F_p\}$,

где W – множество лексем, каждой лексеме сопоставлена информация обо всей совокупности ее форм;

P – множество многословных терминов, описываемых парой <N-грамма, тип структуры>, где N-грамма задает последовательность лексем, а тип структуры определяет вершину и правила согласования элементов N-граммы;

M – морфологическая модель языка, включающая описание морфологических классов и признаков;

G – множество правил согласования для извлечения многословных терминов;

S – проблемно-ориентированная система признаков для разметки терминов;

$F_w = W \to 2^{M \times S}$, $F_p = P \to 2^{G \times S}$ – функции, сопоставляющие терминам наборы признаков.

Особенностью морфологического представления в рамках системы является возможность ее изменения в зависимости от задачи, решаемой с помощью словаря. Пользователь может сформировать собственный набор признаков, классов и обеспечить их интеграцию посредством правил сопоставления с базовым морфологическим представлением. Потребность в изменении классов возникает достаточно редко, например, когда используются дополнительные специализированные словари терминов (словари имен, географических названий) или необходимо включить в словарь слова другого языка.

Морфологический класс определяется частью речи, набором лексических признаков (например, одушевленность или род у существительных) и типом парадигмы.

Морфологический атрибут – это пара $<t^m, V^m>$, где t^m задает тип признака (имя), а V^m – множество значений (например, <Род, {муж, жен, ср}>). Атрибуты в рамках каждого класса делятся на словообразующие или лексические, присущие всем формам лексем данного класса, и словоизменительные, различающие формы одной лексемы.

Для описания словоизменения используется понятие парадигмы – совокупности всех словоформ, относящихся к одной лексеме и имеющих разные грамматические значения. Тип парадигмы задает упорядоченный набор изменяемых атрибутов лексем данного морфологи-

ческого класса (например, для обычного прилагательного это тройка <падеж, число, род>) и функции для вычисления значений признаков для заданной словоформы (точнее, ее флексии) и наоборот. Таким образом, каждой лексеме сопоставляется морфологический класс и парадигма из таблицы парадигм, а для каждой парадигмы можно вычислить значения морфологических признаков относительно типа парадигмы, определяемого морфологическим классом.

Другой важной особенностью системы является поддержка многословных терминов – словосочетаний, сформированных по правилам, реализующих поверхностно-синтаксический анализ. Большинство многословных терминов включают от двух до четырех слов и формируются с помощью правил вида П+С (*аналоговый датчик*) – согласование существительного и прилагательного, С+Срд (*автор учебника*), П+П+С (*новая информационная технология*), С+Прд+Срд (*обработка естественного языка*) и т.п. Встречаются также термины с более сложной структуры, например, с зависимыми предложными группами С+Предл+С (*резервуар с жидкостью*), С+Предл+С+С (*поиск в пространстве состояний*) и т.д.

3.2. Система проблемно-ориентированных признаков

В зависимости от решаемой проблемы термины словаря могут снабжаться различными типами признаков: статистическими (для решения задач классификации), жанровыми (для жанрового анализа текста), семантическими (для семантического анализа), формальными (для выявления маркеров определенных структур) и др.

Статистические признаки накапливают информацию о частоте появления термина в обрабатываемых текстах. Для решения задач классификации текста необходимо, чтобы пользователь задал систему связанных между собой тем, а также наличие обучающего корпуса текстов, т.е. корпуса, размеченного заданными признаками. В словаре для каждого термина хранится: встречаемость в обучающей выборке (абсолютная частота) и количество текстов, в которых встречался термин (текстовая частота), список тем, в которых встретился термин, частота и текстовая частота по каждой теме из списка. Часть параметров (относительная частота, TF*IDF, вес) вычисляется динамически.

Система признаков, используемая для разметки терминов словаря, формируется пользователем и зависит от решаемой задачи, т.е. является проблемно-ориентированной. Для кодирования различной информации о слове (семантической, жанровой, стилистической и др.) предусмотрены следующие возможности.

Класс. Термин может быть отнесен к определенному признаку-классу. Иерархия классов позволяет отнести термин к определенному уровню иерархии, более общему или конкретному с наследованием свойств общего класса.

Атрибут. Для представления лексического значения термина используются атрибуты. Совокупность семантических значений атрибутов, приписанных слову, в определенной мере моделирует компонентную семантическую структуру слова. Основные компоненты семантической структуры термина могут рассматриваться как тезаурусные дескрипторы.

Механизм фиксации значений. Набор признаков (классов, атрибутов и их значений) позволяет достаточно полно описать лексико-семантическое значение термина. Наличие нескольких альтернативных групп позволяет выразить неоднозначность термина.

Рассмотрим семантическую разметку многозначного термина *нефтедобыча* (рис. 2).

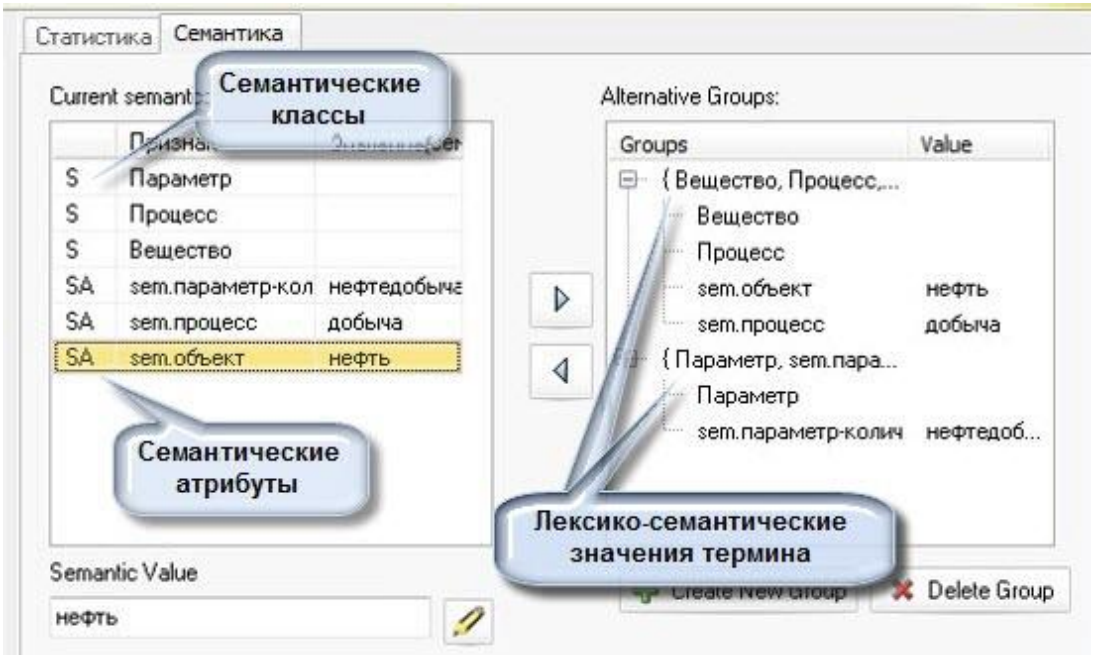


Рис. 2. Разметка термина *нефтедобыча*

При разметке термина вначале фиксируются все возможные признаки термина, а затем осуществляется фиксация значений. Такой механизм позволяет задавать как альтернативные, так и синкретически выраженные значения терминов. Так, в примере, приведенном на рис. 2, термину приписано два лексико-семантических значения: а) процесс *Добыча* над *Веществом* и б) параметр *нефтедобыча*, значение которого может выражаться числовым значением. Дополнительно значения снабжены семантическими атрибутами, позволяющими задать канонические названия признаков.

4. Методы поддержки работы с корпусом текстов

Созданная среда включает словарные компоненты и обработчики, которые обеспечивают, с одной стороны, автоматизацию создания, наполнения, редактирования словарей, с другой – применение созданных словарей для лексического анализа текста и последующую работу со словарной информацией найденных в тексте терминов. Важной частью поддержки пользователя является набор поисковых средств, таких как сортировка и фильтрация терминов по различным наборам параметров, визуализация покрытия текста терминами словаря, построение конкордансов терминов с расширяемым окном контекста и др.

4.1. Обучение словаря на основе корпуса текстов

Процесс извлечения терминологии включает такие этапы, как: a) графематический анализ, обеспечивающий токенизацию и выделение нетекстовых элементов и иноязычных вкраплений, b) лексико-морфологический анализ (лемматизация, определение лексико-грамматических признаков, представление парадигмы, нормализация), c) выделение терминоподобных словосочетаний (идентификация на основе предопределенных грамматических моделей и нормализация), d) обновление статистики найденных терминов.

Для создания словарей используются следующие модули.

Морфологический анализ осуществляется на базе модуля Диалинг (`www.aot.ru`), который содержит универсальный словарь русского языка и обеспечивает поиск слова в словаре, определение его грамматических признаков и нормальной формы. Также поддерживается функция предсказания [6], которая по незнакомому слову формирует гипотезы (как правило, около трех вариантов) о его части речи, нормальной форме и других признаках.

Модуль сборки словокомплексов – извлекает из текста словосочетания по фиксированному набору грамматических правил. Основной задачей модуля является выявление наиболее важных терминообразующих синтаксических групп, большинство из которых представляют собой именные группы либо строятся на их основе.

Результатом обработки корпуса текста с помощью данных модулей является лексическое наполнение словаря с одновременным сбором статистики встречаемости. Если корпус размечен признаками, то соответствующие термины снабжаются данными признаками и относительно признаков также ведется статистика.

Таким образом, среда обеспечивает автоматизацию начального наполнения словарей, на основе которого можно проводить дальнейшие исследования.

4.2. Исследование контекстов термина

Конкорданс – традиционный способ изучения корпуса текста. Он дает полный индекс терминов в ближайших и расширенных контекстах, что дает возможность исследователю проверить свою гипотезу относительно функций той или иной лексической единицы. Конкорданс создается для поиска и извлечения образцов целевых единиц. Таким образом, конкорданс осуществляет обратную связь словаря, словарных терминов с корпусом и обеспечивает своего рода лингвистическую разметку на морфологическом и поверхностно-синтаксическом уровне.

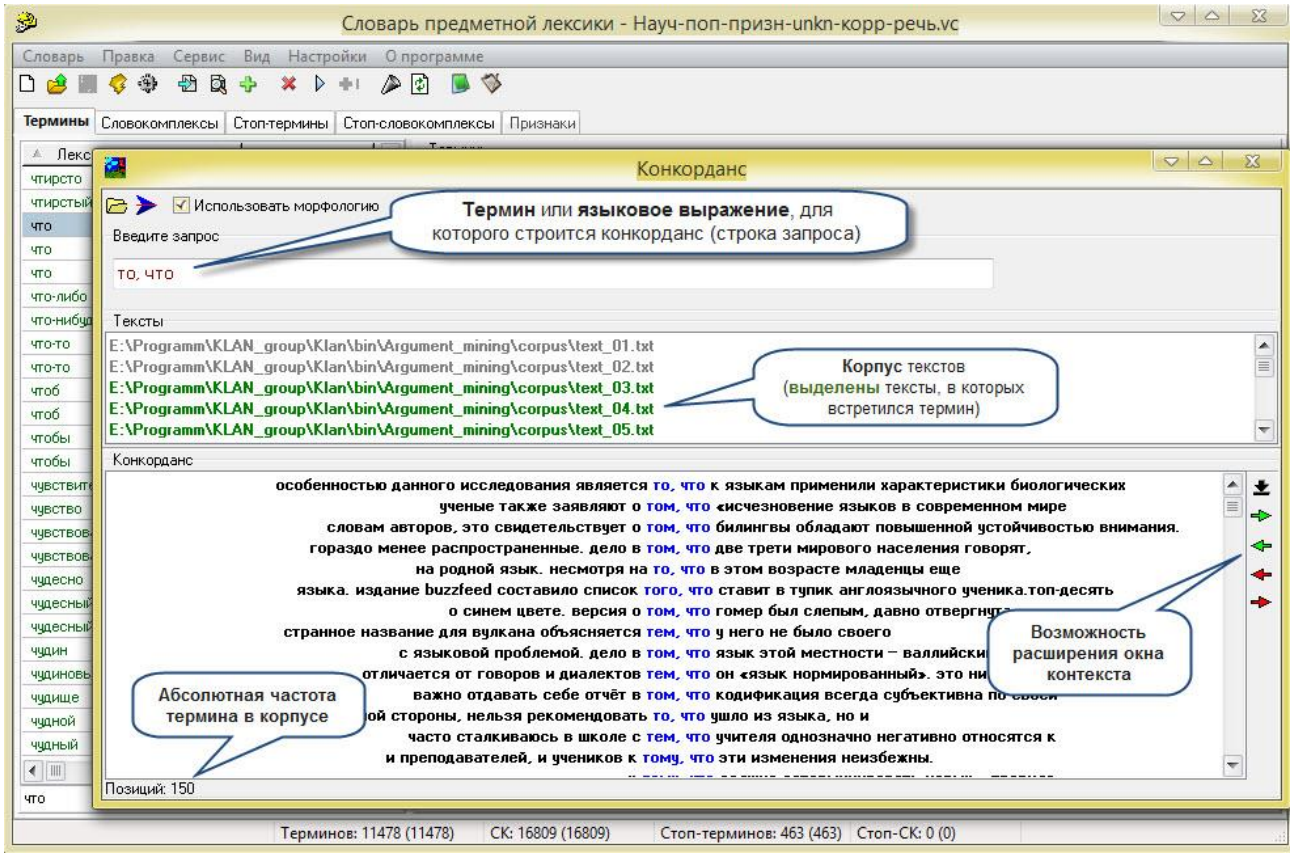


Рис. 3. Исследование контекстов термина с помощью конкорданса

Разработанный модуль конкорданса (рис. 3) работает с небольшими корпусами текстов и осуществляет построение конкордансов в реальном времени, позволяя пользователю уточнять и изменять запрос. При просмотре множества контекстов вхождения термина пользователь может самостоятельно определять длину просматриваемого фрагмента текста (поддерживается пословное расширение контекста, просмотр абзаца или всего сообщения). Так, в приведенном примере вначале был сформирован конкорданс для термина *что* (869 вхожде-

ний), а затем запрос был расширен до языковой конструкции *то, что* (150 вхождений). Более подробный анализ контекста позволил определить, что данная конструкция позволяет выделить аргументацию, представленную косвенной речью эксперта, с точностью ~ 65 %. А при включении в модель описания аргумента речевых и ментальных предикатов и терминов с семантикой человека-эксперта точность распознавания аргументации увеличилась до 86.5 %.

Таким образом, необходимо отметить, что конкорданс является не только средством исследования контекстов терминов, но и важным инструментом проверки гипотез относительно функционирования языка и создания качественных лингвистических ресурсов.

4.3. Методика исследования лексических характеристик текста

В процессе создания лингвистического ресурса можно выделить несколько содержательных этапов (рис. 4): сбор и разметка данных, первоначальное наполнение словаря на основе корпуса, формирование системы признаков для моделирования значений терминов, исследование терминов с помощью конкорданса, генерация гипотез и их экспертная оценка. Полученный ресурс можно использовать для автоматизации разметки новых данных.

Рис. 4. Жизненный цикл разработки лингвистических ресурсов

Процесс первоначального наполнения словаря осуществляет извлечение из массива текстов терминов-кандидатов на базе определенных лингвистических ограничений (морфологических и синтаксических моделей) с использованием технологии обучения по массиву текстов. Морфологический анализ текстов (на основе русского морфологического анализатора) позволяет извлечь однословную лексику. Учитывая, что многие термины специализированных предметных областей универсальному словарю-анализатору неизвестны, используются средства предсказания незнакомых слов. Поверхностный синтаксический анализ выделяет в текстовых сегментах контактные синтаксически связанные группы слов (словокомплексы) на основе фиксированного набора правил.

Система признаков, используемая для разметки текстов, зависит от предметной и проблемной области, её формирование является достаточно сложной научно-исследовательской проблемой. Так, для создания ресурсов, ориентированных на автоматическое извлечение информации или информационный поиск, система признаков может быть построена на основе онтологии предметной области и дополнена необходимыми признаками для решения стандартных лингвистических проблем. Еще одним широко используемым методом построения системы признаков является использование универсальных тезаурусов для фиксации верхнего уровня иерархии дескрипторов с дальнейшим расширением специализированными при-

знаками [1]. В задачах классификации текстов применяются методы для сужения пространства признаков.

Разметка терминов может быть сформирована автоматически при наличии размеченного корпуса текста. В противном случае разметка терминов осуществляется экспертом вручную. Работа эксперта поддерживается инструментальными средствами сортировки и фильтрации терминов по различным наборам признаков, поиска всех многословных терминов, содержащих слово, построения конкорданса терминов с учетом словоизменения. Механизм фиксации значений терминов позволяет достаточно полно описать его особенности.

Формирование более сложных лингвистических моделей и их проверка относительно лексической составляющей позволяет добиться необходимого качества создаваемого ресурса.

5. Заключение

Данная работа посвящена описанию подходов и методов разработки лексикографических ресурсов и проведения корпусных исследований для обеспечения полноты и достоверности разрабатываемых моделей. В фокусе внимания создаваемого инструментария находятся лексические единицы, выступающие в качестве маркеров и индикаторов объектов более высокого уровня (семантического, прагматического, тематического, структурно-жанрового, логико-аргументативного и др.).

Отличительными особенностями рассматриваемого программного комплекса являются возможность его многоцелевого использования при решении различных задач анализа текста, интеграция различных возможностей для проведения языковых исследований, обеспечение настройки словарей на проблемную область пользователя, поддержка мультиязычности.

Развитие предлагаемого подхода заключается в апробации и внедрении методов автоматического построения моделей, ориентированных на извлечении прагматической информации из текстов: описание объектов и их параметров, моделей для извлечения отношений (ситуаций), жанровых особенностей текста, риторической аргументации и т.п.

Литература

1. *Лукашевич Н. В.* Тезаурусы в задачах информационного поиска. М.: МГУ, 2011. 495 с.
2. *Sinclair J.* Corpus, Concordance, Collocation. Edited by Ronald Carter. Oxford: Oxford University Press, 1991, XVIII, 179. 200 p.
3. *Захаров В. П., Хохлова М. В.* Автоматическое выявление терминологических словосочетаний // Структурная и прикладная лингвистика. 2014. Вып. 10. С. 182–200.
4. *Bolshakova E., Loukachevitch N., Nokel M.* Topic Models Can Improve Domain Term Extraction // International conference on Information Retrieval (ECIR-13), Springer Verlag, 2013. LNCS-7814. P. 684–687.
5. *Митрофанова О. А., Захаров В. П.* Автоматизированный анализ терминологии в русскоязычном корпусе текстов // Компьютерная лингвистика и интеллектуальные технологии: тр. межд. конференции «Диалог-2009». С. 321–328.
6. *Сокирко А. В.* Морфологические модули на сайте www.aot.ru // Компьютерная лингвистика и интеллектуальные технологии: тр. межд. конференции Диалог-2004. С. 559–564.

Статья поступила в редакцию 19.07.2019;
переработанный вариант – 05.08.2019.

Сидорова Елена Анатольевна

к.ф.-м.н., с.н.с., лаборатория искусственного интеллекта, Институт систем информатики им. А. П. Ершова СО РАН (630090, Новосибирск, просп. Академика Лаврентьева, 6), тел. (383) 3-307-991, e-mail: lsidorova@iis.nsk.su.

The integrated approach to text lexical characteristics study

E. Sidorova

The integrated approach and software environment for multi-aspect study of the text lexical characteristics are considered. This work is at the junction of corpus linguistics and lexicographical research. The basis of the research is the corpus of text and the problem-oriented dictionary. The proposed environment for supporting the researcher provides tools and interfaces for developing vocabularies and a system of domain features, terms markup, automatic generation of lexical content and accumulation of statistical information, etc. To extract terms the morphological analysis and the construction of phrases based on the rules of matching the grammatical characteristics of words are carried out. To study the contexts of the terms use, concordance construction tools are provided. Concordances allow the researcher to test his or her hypothesis about the functionality of a particular lexical unit. The considered environment allows to solve various text analysis tasks because it integrates various tools for conducting language research and supports customization of vocabularies to a problem area.

Keywords: terminology, domain vocabulary, corpora, concordance.

УДК 004.05

Проектирование сертифицированного компилятора предикатных программ

В. И. Шелехов

Компилятор со средствами дедуктивной верификации подлежит тщательной сертификации для подтверждения высокого уровня доверия к результатам дедуктивной верификации. Разработка компилятора предикатных программ на базе его модели в соответствии с модельно-ориентированной технологией повышает уровень доверия к компилятору. Главной частью модели компилятора является модель внутреннего представления программы, реализуемая в рамках третьего релиза компилятора предикатных программ. В настоящей работе описывается архитектура модели внутреннего представления программы. Важнейшим аспектом модели является анализ типов. В описании модели используется специально разработанный язык спецификации структур данных компилятора. Верификация модели и разработка программы компилятора на базе модели обеспечат высокую надежность компилятора.

Ключевые слова: дедуктивная верификация, сертификация программ, модельно-ориентированная технология.

1. Введение

Метод дедуктивной верификации обеспечивает абсолютную гарантию корректности программы относительно ее спецификации. Ввиду большой сложности и трудоемкости этот метод применяется лишь для критических фрагментов программы в приложениях, где требуется сверхвысокая надежность и безопасность.

Дедуктивная верификация намного проще и быстрее для предикатных программ, чем для аналогичных императивных программ. Преимущество по времени верификации – более чем в 5 раз. Для программы быстрой сортировки с двумя опорными элементами зафиксировано преимущество в 10 раз [1].

В соответствии со стандартами DO178C и ГОСТ Р ИСО/МЭК 15408-3 [2] сертификация программ, для которых применялась дедуктивная верификация, сопровождается сертификацией корректности сопутствующих инструментов и методов. Сертифицируется подсистема верификации вместе с содержащей ее системой программирования, отдельно front-end и back-end генерации кода. Далее в этой цепочке – операционная система, загрузчик программ и, наконец, система команд компьютера. Сертифицируется также сам метод дедуктивной верификации.

Возможность ситуации, когда в процессе компиляции в программу вносится ошибка, принципиально снижает ценность проведенной дедуктивной верификации. Поэтому сертификации компилятора отводится особое внимание. В идеале компилятор должен пройти процедуру дедуктивной верификации. Имеется лишь один такой компилятор. Это оптимизирующий компилятор CompCert [3] с языка Си, доступный с 2015 г. Компилятор состоит из 20 просмотров транслируемой программы и использует 11 промежуточных представлений про-

граммы; для каждого представления разработана формальная семантика. Затраты на дедуктивную верификацию программы компилятора CompCert в системе автоматического доказательства Coq [4] – шесть человеко-лет.

Язык предикатного программирования P [5] принципиально сложнее языка Си. Разработка и дедуктивная верификация компилятора для языка P потребовала бы на порядок больше затрат, что за границей текущих возможностей. Есть другая альтернатива – *модельно-ориентированная* технология. Ее применение при разработке компилятора с языка P позволит сертифицировать компилятор высоким уровнем доверия в соответствии со стандартом ГОСТ Р ИСО/МЭК 15408-3 [2].

Модельно-ориентированная технология в системной и программной инженерии базируется на построении и верификации *модели* создаваемого объекта (или программы). В традиционной технологии разработка большой программной системы сопровождается наличием множественных нестыковок и разломов во внутренних интерфейсах системы. В модельно-ориентированной технологии внутренние интерфейсы выстраиваются строго в соответствии с моделью (после ее верификации), что принципиально повышает надежность программной системы. За последние 20 лет сменилось три поколения модельно-ориентированной технологии – от model-based к model-driven, в которой программа генерируется по модели автоматически, далее – к интегрированной модели, единой для создаваемого объекта и встроенного в него программного обеспечения. В модели выделяются различные слои, например, связанные с надежностью, безопасностью и отказоустойчивостью. Реализуется адекватная интеграция различных слоев.

Примером успешного применения модельно-ориентированной технологии является разработка и верификация модели политики безопасности управления доступом в операционных системах [6], в частности, в Astra Linux Special Edition [7]. На языке EventB разработана формальная модель политик безопасности, доказана ее корректность и согласованность. Подтверждена корректность отображения этой модели на ядро ОС Linux размером 2 млн. строк кода. Проведена дедуктивная верификация центральной компоненты Linux Security Module. Данная работа обеспечила высокий уровень доверия к безопасности Astra Linux Special Edition, подтвержденный международной организацией Linux Foundation[1].

Сверхзадачей настоящей работы является построение полной модели компилятора предикатных программ, включающей:

- модель системы типов языка предикатного программирования;
- модель внутреннего представления транслируемой программы;
- модель оптимизирующих трансформаций предикатной программы;
- модель подсистемы дедуктивной верификации.

В настоящей работе кратко описываются модель типов и модель внутреннего представления. Их полное описание имеется в проекте [8]. Целью разработки моделей является построение компактного корректного описания, в котором разные компоненты модели адекватно согласованы между собой. На базе модели планируется разработать новую версию программы компилятора. В дальнейшем при любых изменениях необходимо будет поддерживать точное соответствие между программой и моделью.

Во втором разделе настоящей работы дается краткое описание языка предикатного программирования P [5]. В следующем разделе представлена концепция построения модели внутреннего представления предикатной программы. Далее описывается модель типов языка P [5]. В заключении суммируются основные результаты и анализируются условия, необходимые для реализации предложенного подхода.

[1] http://www.connect-wit.ru/gk-astra-linux-rasshiryaet-sotrudnichestvo-s-the-linux-foundation.html

2. Язык предикатного программирования

Предикатная программа является предикатом в форме вычислимого оператора H(x: y) с аргументами x и результатами y. Программа имеет две основы: логику и вычислимость, причем логика является первичной, главной. Минимальный полный *базис предикатных программ* определен в виде языка **P0**. Предикатная программа определяется следующей конструкцией:

<имя предиката>(<аргументы>: <результаты>) { <оператор> }.

Пусть x, y и z обозначают разные непересекающиеся наборы переменных. Набор x может быть пустым, наборы y и z не пусты. В составе набора переменных x может использоваться логическая переменная e со значениями **true** и **false**. Пусть B и C – имена предикатов, A и D – имена переменных предикатного типа. Операторами являются: *оператор суперпозиции* B(x: z); C(z: y), *параллельный оператор* B(x: y) || C(x: z), *условный оператор* **if** (e) B(x: y) **else** C(x: y), *вызов предиката* B(x: y) и *оператор каррирования*. В табл. 1 представлен полный базис вычислимых предикатов и соответствующих им операторов.

Таблица 1. Вычислимые предикаты и их программная форма

H(x: y) ≅ ∃z. B(x: z) & C(z: y)	H(x: y) { B(x: z); C(z: y) }
H(x: y, z) ≅ B(x: y) & C(x: z)	H(x: y, z) { B(x: y) || C(x: z) }
H(x: y) ≅ (e ⇒ B(x: y)) & (¬e ⇒ C(x: y))	H(x: y) { **if** (e) B(x: y) **else** C(x: y) }
H(x: y) ≅ B(x~ : y)	H(x: y) { B(x~: y) }
H(A, x: y) ≅ A(x: y)	H(A, x: y) { A(x: y) }
H(x: D) ≅ ∀y,z. D(y: z) ≡ B(x, y: z)	H(x: D) { D(y: z) {B(x, y: z)} }
H(A, x: D) ≅ ∀y,z. D(y: z) ≡ A(x, y: z)	H(A, x: D) { D(y: z) {A(x, y: z)} }

Набор x~ составлен из набора переменных x с возможным добавлением имен предикатных программ. Результатом исполнения оператора каррирования D(y: z) {B(x, y: z)} является новый предикат D, получаемый фиксацией значения набора x.

Для языка **P0** построена формальная операционная семантика ℛ(H)(x, y) и доказано тождество ℛ(H) = H [9]. На базе языка **P0** последовательным расширением и сохранением тождества ℛ(H) = H построен язык предикатного программирования **P** [5]:

$$\mathbf{P_0} \to \mathbf{P_1} \to \mathbf{P_2} \to \mathbf{P_3} \to \mathbf{P_4} = \mathbf{P}\ .$$

Каждый очередной язык в этой цепочке строится на базе предыдущего в виде системы обозначений. Одной из задач построения формальной модели внутреннего представления программы является систематическое построение формальной семантики конечного языка **P** прохождением через все цепочки обозначений.

Система типов вносится в язык **P4**. Предыдущие языки – бестиповые. В языке **P** имеется развитая система типов: подтипы, структуры, множества, алгебраические типы, предикатные типы, массивы. Тип массива является предикатным типом, его значения (массивы) являются тотальными и однозначными предикатами. Типы могут быть параметризованы переменными.

Эффективность предикатных программ достигается применением следующих *оптимизирующих трансформаций* [10], переводящих программу на язык императивного расширения P_{imp}:

- замена хвостовой рекурсии циклом;
- открытая подстановка программы на место ее вызова;
- склеивание переменных: замена всех вхождений одной переменной на другую переменную;

- кодирование объектов алгебраических типов (списков и деревьев) с помощью массивов и указателей.

Далее программа конвертируется на язык Си. В языке **P** нет циклов и указателей, вместо них используются рекурсивные программы и алгебраические типы данных.

3. Модель внутреннего представления программы

Внутреннее представление (ВП) программы, получаемое в результате работы front-end'а[2] компилятора, являясь функционально эквивалентной копией исходной программы, ориентировано на удобство работы с программой в middle- и back-end'ах[3]. Опыт разработки компиляторов показывает, что структура ВП программы обычно громоздка и неоправданно сложна, содержит много дублирующей информации. Около тридцать лет назад было большое число работ по технологии трансляции. В настоящее время разработка компилятора считается рутинной задачей. Однако классическая техника разработки компиляторов не решает проблему построения адекватной структуры ВП. В частности, не следует использовать абстрактное синтаксическое дерево программы в качестве основы для ВП. Основу ВП определяют примитивы исполнения, а не синтаксис.

В настоящей работе описывается метод построения модели ВП, абстрагированной от ее реализации на языке C++. Разрабатываемая модель ВП является частью проекта [8].

3.1. Метаязык

Для описания модели ВП используется язык абстрактных структур, содержащий следующие примитивы:

- произвольные неупорядоченные наборы (множества) объектов;
- упорядоченные последовательности объектов;
- вершины (записи) определенного именованного вида с набором полей (атрибутов);
- абстрактные указатели на объекты;
- понятия, определяющие множество структур некоторого вида.

Например,

Prog(Заголовок программы, Область локализации, Тело программы)

определяет вершину с именем Prog и тремя полями. Значение каждого поля определяется соответствующим понятием.

3.2. Архитектура внутреннего представления программы

Программа во ВП определяется в виде дерева структур. Точнее, это граф с учетом дуг, определяемых абстрактными указателями. Для всех языковых конструкций (операторов, выражений, описаний и др.) используется единое унифицированное представление в виде вершин (записей).

Всякий объект, идентифицируемый некоторым именем в тексте программы, представлен во ВП указателем на описание этого объекта. Описание *именованного объекта* представлено во ВП вершиной определенного вида. Имеются разные виды именованных объектов: простые переменные, типы, предикатные программы, формулы и др.

Совокупность имен и соответствующих им описаний одного уровня объединены в структуру вида *область локализации*, определяющую отображение имен на описания. Об-

[2] front-end реализует синтаксический и семантический анализ программы с построением ее ВП.

[3] back-end – языковый процессор, определяющий вторую часть компилятора, обычно – генератор кода. middle-end – языковый процессор, преобразующий ВП программы.

ласть локализации прикрепляется к некоторой конструкции программы. Например, область локализации, содержащая параметры типа в описании типа, прикрепляется к описанию типа.

Иерархия языковых конструкций определяет иерархию областей локализации, на базе которой проводится *идентификация* – определение соответствующего описания для каждого вхождения имени в программе в процессе семантического анализа. Дополнительно, вне иерархии областей локализации, существуют *автономные области локализации* для полей структур и интерфейсных элементов классов, причем для наследуемых классов реализуется независимая иерархия. Приведенная модель именования объектов является адекватной для проведения идентификации, в том числе и для полиморфных объектов.

Структура операторной части предикатной программы определена в виде дерева сегментов. *Сегмент* – фрагмент программы с одним входом и несколькими выходами. Каждый *выход* связан с некоторым входом другого сегмента либо с выходом программы. Отметим, что все виды операторов выбора (**switch**, **case**) [5] преобразуются в условные операторы. Вызов функции заменяется во ВП вызовом предиката и корректным образом выносится из выражения, содержащего вызов функции. В итоге во ВП программа возвращается к языку $\mathbf{P_2}$.

Во ВП устраняются все умолчания, существующие в языке P [5]. Например, вхождение переменной в позиции значения представляется вершиной Value(Переменная). Устраняется полиморфизм операций. Вместо операции «+» используются вершины вида AddNat, AddInt, AddSet, AddLists и др.

4. Анализ типов

Имена объектов и области локализации не нужны для работы с ВП в middle- и back-end'ах. Они необходимы лишь при отображении программы в исходное текстовое представление, а также при диагностике ошибок, где дополнительно требуются координаты (с точностью до символа) по входному тексту программы. Произвольный объект программы идентифицируется указателем на структуру, представляющую описание этого объекта.

Атрибутами большинства объектов программы являются *типы*. Определение типов объектов в соответствии с правилами языка P [5] и проверка разнообразных ограничений на типы требуют нетривиального анализа программы. Анализ типов (type checking), обычно называемый семантическим анализом программы, является наиболее сложной и трудоемкой частью в реализации front-end'а компилятора.

Модель ВП включает модель обширной системы типов языка P. Допускаются подтипы, определяемые как множество истинности некоторого предиката. Типы могут быть параметризованы переменными. Имеются алгебраические типы, представителями которых являются списки и деревья. Рекурсивные типы определяются через аппарат наименьшей неподвижной точки. Предикатные типы, являясь типами второго порядка, определяют типы аргументов и результатов предикатных программ. Массивы определены как частный случай предикатных типов со свойствами тотальности и однозначности.

Тип может быть параметром предикатной программы. Операции с таким типом также поставляются через параметры. Спецификация типа обычно реализуется независимой теорией в стиле алгебраических спецификаций.

Для типов, операций и предикатных программ допускается полиморфизм. Это существенно осложняет идентификацию и анализ типов. В частности, необходимо проводить качественную аппроксимацию значений переменных предикатного типа.

На типах определяются отношения *тождества*, *совместимости* (один тип является подмножеством другого) и *согласованности* (существование общей мажоранты). Эти виды отношений задействованы в правилах семантики языка P для смежных конструкций, в частности, для операндов бинарных операций. Данные виды отношений определяются индуктивно в виде системы правил.

Типовый терм – языковая конструкция, значением которой является тип. Нетривиальна проблема тождества типовых термов.

5. Заключение

Чтобы обеспечить высокий уровень доверия к результатам дедуктивной верификации предикатных программ, необходимы соответствующие гарантии корректности разрабатываемой программы компилятора с языка P. С этой целью разрабатывается модель ВП программы, на базе которой планируется выстраивать все интерфейсы компилятора.

Цель первого этапа – обеспечить адекватность ВП языку P и согласованность частей ВП. Описание структуры ВП в разд. 3.2 и в проекте [8] – это лишь форма (синтаксис) программы, которую необходимо наполнить формализованной семантикой на базе содержательного описания языка P [5]. Следующий шаг – построение формальной операционной семантики ВП программы. Полная модель ВП должна пройти процесс ее верификации.

Еще одной задачей является построение формальной модели системы типов языка P и ее верификация.

Наиболее сложной задачей следующего уровня является построение модели оптимизирующих трансформаций предикатных программ. Необходимо формально доказать корректность применения всех видов оптимизирующих трансформаций.

Практическое применение модельно-ориентированной технологии требует владения формальными методами. Эта дисциплина более десятка лет преподается в МГУ на факультете МВК и в НГУ на кафедре программирования[4].

Литература

1. *Шелехов В. И., Чушкин М. С.* Верификация программы быстрой сортировки с двумя опорными элементами // Научный сервис в сети Интернет. М.: ИПМ им. М. В. Келдыша, 2018. 26 с. URL: http://persons.iis.nsk.su/files/persons/pages/dqsort.pdf (дата обращения: 12.03.2019).
2. ГОСТ Р ИСО/МЭК 15408-3-2013. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности. М.: Стандартинформ, 2014. 150 с.
3. *Kästner D., Wünsche U., Barrho J., Schlickling M., Schommer B., Schmidt M., Ferdinand C., Leroy X., Blazy S.* CompCert: Practical experience on integrating and qualifying a formally verified optimizing compiler // ERTS 2018: Embedded Real Time Software and Systems. SEE, January 2018. P. 1–9.
4. Coq Proof Assistant [Электронный ресурс]. URL: https://coq.inria.fr/ (дата обращения: 12.03.2019).
5. *Карнаухов Н. С., Першин Д. Ю., Шелехов В. И.* Язык предикатного программирования Новосибирск, 2018. 42 с. URL: http://persons.iis.nsk.su/files/persons/pages/plang14.pdf (дата обращения: 12.03.2019).
6. *Девянин П. Н., Ефремов Д. В., Кулямин В. В., Петренко А. К., Хорошилов А. В., Щепетков И. В.* Моделирование и верификация политик безопасности управления доступом в операционных системах // М.: Горячая линия – Телеком, 2019. 261 с. URL: http://www.ispras.ru/publications/security_policy_modeling_and_verification.pdf (дата обращения: 21.03.2019).
7. Операционные системы Astra Linux [Электронный ресурс]. URL: http://www.astralinux.ru (дата обращения: 12.03.2019).

[4] Видеолекции по формальным методам: http://wasp.iis.nsk.su/

8. *Шелехов В. И.* Проект системы предикатного программирования. Новосибирск: ИСИ СО РАН, 2018. 24 с. URL: http://persons.iis.nsk.su/files/persons/pages/project1.pdf (дата обращения: 12.03.2019).
9. *Шелехов В. И.* Семантика языка предикатного программирования // ЗОНТ-15. Новосибирск, 2015. 13 с. URL: http://persons.iis.nsk.su/files/persons/pages/semZont1.pdf (дата обращения: 12.03.2019).
10. *Каблуков И. В., Шелехов В. И.* Реализация оптимизирующих трансформаций в системе предикатного программирования // Системная информатика. 2017. № 11. С. 21–48. Электрон. журн. 2018. URL: http://persons.iis.nsk.su/files/persons/pages/opttransform4.pdf (дата обращения: 12.03.2019).

Статья поступила в редакцию 24.07.2019;
переработанный вариант – 26.08.2019.

Шелехов Владимир Иванович

к.т.н., зав. лаб. системного программирования, Институт систем информатики имени А. П. Ершова СО РАН (630090, Новосибирск, просп. Ак. Лаврентьева, 6), доцент кафедры программирования НГУ, тел. (383) 330-27-21, e-mail: vshel@iis.nsk.su.

Certified compiler design of predicate programs

V. Shelekhov

A compiler with deductive verification facilities should be thoroughly certified to obtain high level of assurance. The development of the compiler for a predicate program using a compiler model in accordance with a model-based design method makes the compiler to be of a higher level of assurance. The main part of a compiler model is the model of inner program representation for a predicate program. The model is designed in the framework of third release of the compiler. The model architecture for inner representation of predicate program is described. Type checking is specially analyzed as an important aspect in the compiler development. The verification of the model and compiler development on the base of the model should guarantee a high level of compiler reliability.

Keywords: deductive verification, program certification, model-based design.