ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн 11 2015

Учредитель: Издательство "НОВЫЕ ТЕХНОЛОГИИ"

Издается с сентября 2010 г.

Редакционный совет
Садовничий В.А., акад. РАН (*председатель*)
Бетелин В.Б., акад. РАН
Васильев В.Н., чл.-корр. РАН
Жижченко А.Б., акад. РАН
Макаров В.Л., акад. РАН
Панченко В.Я., акад. РАН
Стемпковский А.Л., акад. РАН
Ухлинов Л.М., д.т.н.
Федоров И.Б., акад. РАН
Четверушкин Б.Н., акад. РАН

Главный редактор
Васенин В.А., д.ф.-м.н., проф.

Редколлегия
Антонов Б.И.
Афонин С.А., к.ф.-м.н.
Бурдонов И.Б., д.ф.-м.н., проф.
Борзовс Ю., проф. (Латвия)
Гаврилов А.В., к.т.н.
Галатенко А.В., к.ф.-м.н.
Корнеев В.В., д.т.н., проф.
Костюхин К.А., к.ф.-м.н.
Липаев В.В., д.т.н., проф.
Махортов С.Д., д.ф.-м.н., доц.
Манцивода А.В., д.ф.-м.н., доц.
Назиров Р.Р., д.т.н., проф.
Нечаев В.В., д.т.н., проф.
Новиков Б.А., д.ф.-м.н., проф.
Павлов В.Л. (США)
Пальчунов Д.Е., д.ф.-м.н., доц.
Петренко А.К., д.ф.-м.н., проф.
Позднеев Б.М., д.т.н., проф.
Позин Б.А., д.т.н., проф.
Серебряков В.А., д.ф.-м.н., проф.
Сорокин А.В., к.т.н., доц.
Терехов А.Н., д.ф.-м.н., проф.
Филимонов Н.Б., д.т.н., проф.
Шапченко К.А., к.ф.-м.н.
Шундеев А.С., к.ф.-м.н.
Щур Л.Н., д.ф.-м.н., проф.
Язов Ю.К., д.т.н., проф.
Якобсон И., проф. (Швейцария)

Редакция
Лысенко А.В., Чугунова А.В.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн “Сириус"

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

© Издательство "Новые технологии", "Программная инженерия", 2015

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 11
November
2015

Published since September 2010

Editorial Council:

SADOVNICHY V. A., Dr. Sci. (Phys.-Math.), Acad. RAS (*Head*)
BETELIN V. B., Dr. Sci. (Phys.-Math.), Acad. RAS
VASIL'EV V. N., Dr. Sci. (Tech.), Cor.-Mem. RAS
ZHIZHCHENKO A. B., Dr. Sci. (Phys.-Math.), Acad. RAS
MAKAROV V. L., Dr. Sci. (Phys.-Math.), Acad. RAS
PANCHENKO V. YA., Dr. Sci. (Phys.-Math.), Acad. RAS
STEMPKOVSKY A. L., Dr. Sci. (Tech.), Acad. RAS
UKHLINOV L. M., Dr. Sci. (Tech.)
FEDOROV I. B., Dr. Sci. (Tech.), Acad. RAS
CHETVERTUSHKIN B. N., Dr. Sci. (Phys.-Math.), Acad. RAS

Editor-in-Chief:

VASENIN V. A., Dr. Sci. (Phys.-Math.)

Editorial Board:

ANTONOV B.I.
AFONIN S.A., Cand. Sci. (Phys.-Math)
BURDONOV I.B., Dr. Sci. (Phys.-Math)
BORZOVS JURIS, Dr. Sci. (Comp. Sci), Latvia
GALATENKO A.V., Cand. Sci. (Phys.-Math)
GAVRILOV A.V., Cand. Sci. (Tech)
JACOBSON IVAR, Dr. Sci. (Philos., Comp. Sci.), Switzerland
KORNEEV V.V., Dr. Sci. (Tech)
KOSTYUKHIN K.A., Cand. Sci. (Phys.-Math)
LIPAEV V.V., Dr. Sci. (Tech)
MAKHORTOV S.D., Dr. Sci. (Phys.-Math)
MANCIVODA A.V., Dr. Sci. (Phys.-Math)
NAZIROV R.R. , Dr. Sci. (Tech)
NECHAEV V.V., Cand. Sci. (Tech)
NOVIKOV B.A., Dr. Sci. (Phys.-Math)
PAVLOV V.L., USA
PAL'CHUNOV D.E., Dr. Sci. (Phys.-Math)
PETRENKO A.K., Dr. Sci. (Phys.-Math)
POZDNEEV B.M., Dr. Sci. (Tech)
POZIN B.A., Dr. Sci. (Tech)
SEREBRJAKOV V.A., Dr. Sci. (Phys.-Math)
SOROKIN A.V., Cand. Sci. (Tech)
TEREKHOV A.N., Dr. Sci. (Phys.-Math)
FILIMONOV N.B., Dr. Sci. (Tech)
SHAPCHENKO K.A., Cand. Sci. (Phys.-Math)
SHUNDEEV A.S., Cand. Sci. (Phys.-Math)
SHCHUR L.N., Dr. Sci. (Phys.-Math)
YAZOV Yu. K., Dr. Sci. (Tech)

Editors: LYSENKO A.V., CHUGUNOVA A.V.

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/pi.html, e-mail:prin@novtex.ru

УДК 519.683.2

В. И. Шелехов, канд. техн. наук, зав. лаб., e-mail: vshel@iis.nsk.su,
Институт систем информатики им. А. П. Ершова, г. Новосибирск

Оптимизация автоматных программ методом трансформации требований

Технология автоматного программирования ориентирована на разработку простых, надежных и эффективных программ для класса реактивных систем. Автоматная программа реализует конечный автомат в виде гиперграфа управляющих состояний. В качестве языка спецификаций автоматных программ предложен язык продукций для описания функциональных требований. Построение эффективных автоматных программ реализовано применением набора оптимизирующих трансформаций программы, представленной в виде набора требований. Данный метод проиллюстрирован на примере сложного протокола передачи данных ATM Adaptation Layer уровня Type 2 AAL.

Ключевые слова: *понимание программ, автоматное программирование, определение требований, продукционная система искусственного интеллекта, уровень адаптации ATM*

Введение

Автоматная программа определяется в виде конечного автомата и состоит из нескольких *сегментов кода*. Вершина автомата — *управляющее состояние* программы. Ориентированная гипердуга автомата соответствует некоторому сегменту кода и связывает одну вершину с одной или несколькими другими вершинами [1]. Сегменты кода конструируются из фрагментов, являющихся предикатными программами [2].

Автоматные программы принадлежат классу программ-процессов (реактивных систем), более сложному по сравнению с классом программ-функций. Технология построения надежных и эффективных программ-функций разработана в рамках исследований по предикатному программированию [2—5]. Технология автоматного программирования [1, 6] интегрирована с технологиями предикатного и объектно-ориентированного программирования. Гиперграфовая структура автоматной программы обеспечивает высокую гибкость, выразительность и эффективность программ.

Наряду с операторным языком автоматных программ [1] используется язык спецификаций требований. *Требования* — совокупность утверждений относительно свойств разрабатываемой программы. Одним из видов требований являются *функциональные требования*, определяющие поведение программы. Их наиболее популярной формой являются *сценарии использования* (*use case*) [7]. В описываемом в данной статье подходе сценарии использования представлены в виде правил на языке продукций [8], обычно применяемом для систем искусственного интеллекта. Это простой язык с высокой степенью декларативности. Спецификация на этом языке компактна и легко транслируется в автоматную программу, что позволяет использовать язык спецификации требований как язык автоматного программирования.

Типовые методы разработки автоматных программ представлены на примерах в работе [6]. Существуют автоматные программы, например, протоколы, для которых потери эффективности недопустимы. Такие программы обычно сложны. Их построение является трудной задачей. В настоящей статье предложен метод построения сложных автоматных программ применением трансформаций, реализующих процесс последовательного улучшения программы, начиная с простой неэффективной программы, представленной в виде набора требований. Применены следующие трансформации: эквивалентные замены, специализация и др. Методы построения и оптимизации автоматных программ проиллюстрированы на примере сложного протокола передачи данных ATM Adaptation Layer уровня Type 2 AAL [9].

Обзор работ

В мировой практике технологию определения требований разрабатывают и применяют в основном для больших интернетовских информационных систем и систем телекоммуникации, а также в системной инженерии (системотехнике). Технология спецификации требований отражена в стандарте [10]. В предлагаемом в данной статье подходе определение требований рассматривается для всего класса реактивных систем.

Инженерия требований имеет длительную, почти сорокалетнюю историю, в том числе и в нашей стра-

не, в основном на предприятиях аэрокосмического комплекса. К сожалению, исследования в этом направлении велись независимо и слабо интегрированы с мировым опытом.

Языками спецификации требований являются следующие: естественный язык — английский язык (80 % случаев), формализованное подмножество естественного языка с использованием аппарата онтологий (15 % случаев) и формальный язык (5 %) [11]. Популярной проблематикой является трансляция требований с естественного языка на формальные языки. Формальными языками требований являются: RSML [12], Statechart [13] SDL [14], UML и др. Большинство из них являются графическими. В работе [15] выявлены существенные недостатки языка UML при его использовании для спецификации требований производственных систем. Формальными языками спецификации требований являются также общеизвестные универсальные языки спецификаций, такие как VDM, Z, B и др. Семейство темпоральных языков спецификаций также используется для спецификации требований программных систем, в частности, систем реального времени [16]; наиболее популярным для спецификации требований является язык LTL, см., например [17]. Язык определения требований, предложенный в настоящей работе, существенно отличается от перечисленных языков. Наиболее близок к нему разработанный в целях тестирования язык спецификаций реактивных систем [18].

Продукционные системы искусственного интеллекта [8] реализуются набором правил вида <условие> → <действие>. Правила именно такого вида используются в настоящей работе для определения требований. Ранее язык продукционных правил не применялся для определения требований программных систем. Исключением является работа [19], где язык продукций использован для разработки гибридных систем, однако без осознания того, что продукции являются требованиями к разрабатываемой системе. Другой разновидностью продукционных правил являются *охраняемые действия* (*guarded actions*). Языками охраняемых действий являются**:** язык универсального внутреннего представления для нескольких разнообразных языков спецификаций в целях верификации, моделирования и синтеза [20], а также язык описания аппаратуры BlueSpec [21].

Метод трансформации требований используют в процессе построения требований от наиболее общих к детальным [22]. Его применяют лишь в системной инженерии. Трансформации делят на два класса: сверху вниз (добавление новых элементов) и снизу вверх (связывание существующих элементов). Определены следующие виды трансформаций: декомпозиция, специализация, уточнение целей (условий), агрегация. Других работ по трансформации требований на формальных языках не найдено. Трансформации неформальных требований на естественном языке и трансформации требований в дизайны программ не имеют ничего общего с трансформациями, используемыми в настоящей работе.

Базис автоматного программирования

Понятие автоматной программы. *Автоматная программа* определяется в виде конечного автомата и состоит из нескольких *сегментов кода*. Вершина автомата соответствует некоторому *управляющему состоянию*. Ориентированная гипердуга автомата соответствует некоторому сегменту кода и связывает одну вершину с одной или несколькими другими вершинами. В качестве примера автоматной программы рассмотрим модуль операционной системы (ОС), реализующий сценарий работы с пользователем (рис. 1).

В управляющем состоянии `login` ОС запрашивает имя пользователя. Если полученное от пользователя имя существует в системе, она переходит в управляющее состояние `passw`, иначе возвращается в состояние `login`. В состоянии `passw` система запрашивает пароль. Если поданная пользователем строка соответствует правильному паролю, то пользователь допускается к работе и система переходит в состояние `session`. При завершении работы пользователя система переходит в состояние `login`. Отметим, что реальная программа взаимодействия ОС с пользователем существенно сложнее; в частности,

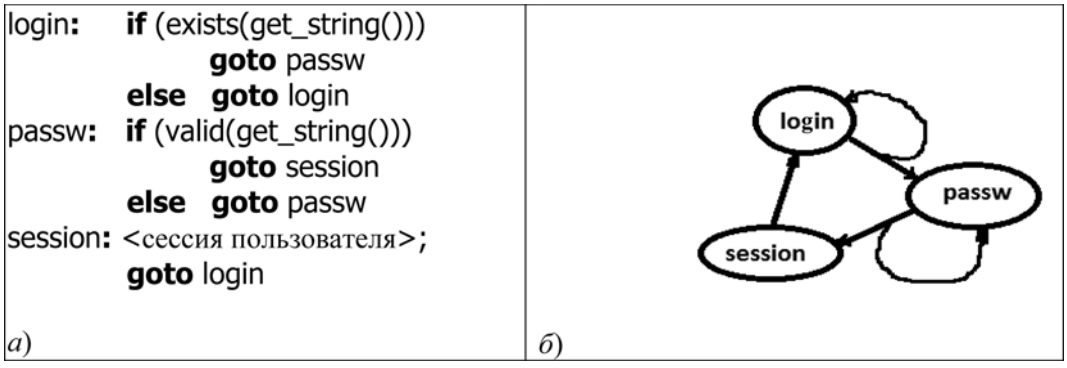


Рис. 1. Схема работы ОС с пользователем:
а — программа; *б* — автомат программы

функция `get _ string` должна поставлять текст в зашифрованном виде.

Состояние автоматной программы определяется значениями набора переменных, модифицируемых в программе, за исключением локальных переменных. Взаимодействие с *внешним окружением* автоматной программы реализуется через прием и посылку *сообщений*, а также через *разделяемые переменные*, доступные в данной программе и других программах из ее окружения. Операторы *ввода* и *вывода* рассматриваются как упрощенная форма операторов посылки и приема сообщений.

Классы программ. Генеральная классификация определяет два класса программ: невзаимодействующие программы (или *программы-функции*) и *программы-процессы*. Данные два класса составляют более 90 % всех программ. Существуют другие, более сложные классы, например языковые процессоры и операционные системы; они находятся на метауровне по отношению к первым двум классам. Языковые процессоры — это интерпретаторы программ, компиляторы, оптимизаторы, трансформаторы и т. д. Приведенная классификация не покрывает всего спектра программ и различных особенностей, например таких, как тесная интеграция программы с данными.

Класс невзаимодействующих программ или класс программ-функций. Программа принадлежит этому классу, если она не взаимодействует с внешним окружением. Точнее, если возможно перестроить программу таким образом, что все операторы ввода данных находятся в начале программы, а весь вывод собран в конце программы, то такая программа относится к классу невзаимодействующих программ. Программа обязана всегда завершаться, поскольку бесконечно работающая и невзаимодействующая программа бесполезна. Следовательно, программа определяет функцию, вычисляющую по набору входных данных (аргументов) некоторый набор результатов. Класс программ-функций, по меньшей мере, содержит программы для задач дискретной и вычислительной математики.

Класс программ-процессов (автоматных программ). Любая программа-процесс является *реактивной системой*, реализующей взаимодействие с внешним окружением программы и реагирующей на определенный набор событий (сообщений) в окружении программы. Определим структуру класса программ-процессов, т. е. класса реактивных систем.

В общем случае программа-процесс определяется в виде композиции нескольких автоматных программ, исполняемых параллельно и взаимодействующих между собой через сообщения и разделяемые переменные. Каждая из параллельно исполняемых программ определяется независимым автоматом.

Подклассом реактивных систем являются *гибридные системы*, соединяющие дискретное и непрерывное поведение. Часть переменных состояния гибридной системы соответствует непрерывным параметрам (типа **`real`**), изменение которых реализуется независимо от программы гибридной системы по определенным законам, обычно формулируемым в виде дифференциальных уравнений. Важнейшими подклассами гибридных систем являются контроллеры систем управления и временные автоматы.

Система управления реализует взаимодействие с *объектом управления* для поддержания его функционирования в соответствии с поставленной целью. Системы управления применяют в аэрокосмической отрасли, энергетике, медицине, массовом транспорте и других отраслях. На каждом шаге вычислительного цикла *контроллер системы управления* получает входную информацию из окружения и обрабатывает ее. По результатам вычислений генерируется управляющий сигнал для воздействия на объект управления. Типовая структура контроллера системы управления определена в работе [23].

Временной автомат реализует функционирование процесса, используя показания времени. Пересчет времени проводится вне автоматной программы (временного автомата). Рассматривают различные модели автоматов с дискретным и непрерывным временем [24]. Временной автомат является *системой реального времени*, если взаимодействие с окружением должно удовлетворять временным ограничениям. В системах с жестким реальным временем непредоставление результатов вычислений к определенному сроку является фатальной ошибкой. Большинство систем управления являются встроенными системами.

Автоматная программа является *детерминированной*, если из каждой вершины автомата (управляющего состояния) исходит не более одной гипердуги. Для *недетерминированного автомата* допускается несколько гипердуг (сегментов кода), исходящих из одного управляющего состояния. При исполнении программы из данного управляющего состояния недетерминированно выбирается один из сегментов кода. Недетерминированный автомат становится *вероятностным*, если для каждой гипердуги определена вероятность ее выбора. Отметим, что недетерминизм неявно реализуется для параллельной композиции автоматных программ.

Автоматная программа может быть составлена из частей, принадлежащих разным подклассам реактивных систем. Например, возможно сочетание вероятностных и недетерминированных автоматов в рамках одной программы. Системы реального времени в большинстве случаев являются системами управления. В дополнение к этому автоматная программа может быть частью *распределенной системы*. Отметим также возможность интеграции с объектно-ориентированной технологией, когда состояние автоматной программы реализовано как объект класса.

Предикатное программирование. Предикатная программа относится к классу программ-функций и является предикатом в форме вычислимого оператора. Язык предикатного программирования P [2]

обладает большей выразительностью в сравнении с языками функционального программирования.

Предикатная программа состоит из набора рекурсивных программ на языке Р следующего вида:

```
<имя программы>(<описания аргументов>:
                    <описания результатов>)
pre <предусловие>
{ <оператор> }
post <постусловие>
```

Необязательные конструкции предусловия и постусловия являются формулами на языке исчисления предикатов; они используются для улучшения понимания программ и для дедуктивной верификации [2—5]. Ниже представлены основные конструкции языка Р: оператор присваивания, блок (оператор суперпозиции), параллельный оператор, условный оператор, вызов программы и описание переменных, используемое для аргументов, результатов и локальных переменных.

```
<переменная> = <выражение>
{<оператор1>; <оператор2>}
<оператор1> || <оператор2>
if (<логическое выражение>) <оператор1>
                          else <оператор2>
<имя программы>(<список аргументов>:
                      <список результатов>)
<тип> <пробел> <список имен переменных>
```

В предикатном программировании запрещены такие языковые конструкции, как циклы и указатели, серьезно усложняющие программу. Вместо циклов используют рекурсивные функции, а вместо массивов и указателей — объекты алгебраических типов (списки и деревья).

Эффективность предикатных программ достигается применением следующих оптимизирующих трансформаций, переводящих программу на императивное расширение языка Р:

- замена хвостовой рекурсии циклом;
- подстановка тела программы на место ее вызова;
- склеивание переменных — замена всех вхождений одной переменной на другую переменную;
- кодирование алгебраических типов (списков и деревьев) с помощью массивов и указателей.

Эффективность программы также обеспечивается оптимизацией, реализуемой программистом, на уровне предикатной программы. Для приведения рекурсии к хвостовому виду применяют метод обобщения исходной задачи. Далее обычно открывается возможность проведения серии последующих улучшений алгоритма. Итоговая программа по эффективности не уступает написанной вручную и, как правило, короче [2—5]. Отметим, что в функциональном программировании (при общеизвестной ориентации на предельную компактность и декларативность [25]) оптимизация программы полностью возлагается на транслятор, в частности, обеспечивается автоматическое приведение рекурсии к хвостовому виду. Разумеется, функциональное программирование существенно уступает в эффективности, поскольку даже применением изощренных методов оптимизации невозможно автоматически воспроизвести серию оптимизаций, совершаемых программистом вручную.

Гиперфункции. Рассмотрим предикатную программу следующего вида:

```
pred A(x: y, z, c)
pre P(x)
{ ... }
post c=C(x) & (C(x) ⇒S(x, y)) & (¬C(x) ⇒R(x, z));
```

Здесь x, y и z — непересекающиеся возможно пустые наборы переменных; P(x), C(x), S(x, y) и R(x, z) — логические утверждения. Предположим, что все присваивания вида c = **true** и c = **false** — последние исполняемые операторы в теле предиката. Программа A может быть заменена следующей программой в виде *гиперфункции*:

```
hyp A(x: y #1: z #2)
pre P(x) pre 1: C(x)
{ ... }
post 1: S(x, y) post 2: R(x, z);
```

В теле гиперфункции каждое присваивание c = **true** заменено оператором перехода #1, а c = **false** — #2.

Гиперфункция A имеет две *ветви* результатов: первая ветвь включает набор переменных y, вторая ветвь — z. *Метки* 1 и 2 — дополнительные параметры, определяющие два различных *выхода* гиперфункции. *Спецификация гиперфункции* состоит из двух частей. Утверждение после **pre** 1 есть предусловие первой ветви; предусловие второй ветви — просто отрицание предусловия первой ветви. Утверждения после **post** 1 и **post** 2 есть постусловия для первой и второй ветвей соответственно.

Использование гиперфункций делает программу короче, быстрее и проще для понимания [5, 26]. Отметим, что модель программ-процессов в форме гиперграфа является естественным продолжением аппарата гиперфункций.

Язык требований. В качестве языка спецификаций программ-процессов предлагается язык продукций [8], применяемый для описания сценариев использования [7] — одного из видов функциональных требований. Это простой язык с высокой степенью декларативности, характерной для языков логического программирования. Он позволяет писать компактные спецификации программ-процессов. Спецификация автоматной программы в виде набора правил легко транслируется в эффективную

автоматную программу, что позволяет использовать язык спецификации требований как язык автоматного программирования.

Требование определяет один из вариантов функционирования автоматной программы и имеет следующую структуру:

```
<условие1>, <условие2>,..., <условиеn> →
                <действие1>, ..., <действиеm>
```

Условиями являются: логические выражения, управляющие состояния, получаемые сообщения. *Действиями* являются: простые операторы, вызовы программ, посылаемые сообщения и итоговые управляющие состояния. Требование является спецификацией некоторого сегмента кода или его части. Семантика требования следующая: если в данный момент времени истинны все условия в левой части требования, то последовательно исполняется набор действий в правой части. Формальная семантика строится на базе темпоральной логики: условия определены для текущего управляющего состояния, а результаты действий — для следующего.

Управляющее состояние в качестве <условия> есть утверждение того, что исполнение программы находится в данном управляющем состоянии. Оно должно быть первым в списке условий и может быть опущено лишь в случае, когда автоматная программа имеет единственное управляющее состояние. Управляющее состояние в качестве <действия> — это следующее управляющее состояние, с которого продолжится исполнение программы после завершения исполнения данного требования. Оно должно быть последним в списке действий.

Неблокированный прием сообщения реализуется конструкцией `<имя сообщения>(<параметры>)`. Ее значение — **true**, если из окружения получено сообщение с указанным именем. Оператор

```
receive <имя сообщения>(<параметры>)
```

реализует прием сообщения с ожиданием появления сообщения из окружения. Отметим, что конструкция вида `M: if (<сообщение>) <оператор> else #M` эквивалентна

```
receive <сообщение>; <оператор>.
```

Оператор `#M` есть оператор **goto** `M`. Оператор **send** `m(e)` посылает сообщение `m` с параметрами — значениями набора выражений `e`.

В качестве примера рассмотрим требования для модуля, реализующего сценарий работы ОС с пользователем (см. рис. 1).

Содержательное описание представлено выше.
Окружение.

```
message Get(string str);
            // получение строки от пользователя
```

Управляющие состояния: `login, passw, session;`
Требования.

```
login, Get(str) → User(str: #login: #passw);
passw, Get(str) → Password(str: #passw: #session);
session → UserSession(), login.
```

Здесь `User` и `Password` — гиперфункции с двумя ветвями без результирующих переменных.

Тип **time** используется для переменных и констант, значениями которых являются показания времени. Оператор **set** `t`, эквивалентный оператору **time** `t = 0`, реализует установку таймера переменной `t`. Ее изменение проводится непрерывно некоторым механизмом, не зависящим от автоматной программы.

Методы оптимизации автоматных программ. В типичном случае при построении автоматных программ оптимизация не требуется. Типовые методы разработки автоматных программ без оптимизации представлены на примерах в работе [6]. Существуют автоматные программы, например, протоколы, для которых потери эффективности недопустимы. Для построения сложных автоматных программ применяют оптимизирующие трансформации, реализующие процесс последовательного улучшения программы, начиная с простой неэффективной программы. Ниже описаны основные трансформации и связанные с ними особенности оптимизации автоматных программ.

Оператор суперпозиции `B(x: y); C(y: z)`, условный оператор **if** `(e(x)) B(x: y)` **else** `C(x: y)` и параллельный оператор `B(x: y) || C(x: z)` определяют основные формы конструирования алгоритмов в предикатном программировании. Эти же формы конструирования применимы и в автоматном программировании с преломлением на структуру автоматной программы.

Аналогом условного оператора является фрагмент автоматной программы

```
H1: inv e(x); B(x: y) #H3
H2: inv ¬e(x); C(x: y) #H3
H3:
```

Инвариант **inv** e(x) ассоциирован с управляющим состоянием H1 и должен быть истинным, когда исполняемая программа приходит в состояние H1. Инвариант не вычисляется и должен быть истинным априори. Прежде всего, инвариант — средство конструирования программы, хотя он может быть использован и для верификации программы.

Следует особо подчеркнуть, что инварианты автоматной программы принципиально отличаются от инвариантов циклов и инвариантов классов императивной программы. В типичном случае, когда не требуется оптимизации автоматной программы, управляющее состояние не содержит инварианта; иначе говоря, инвариант тождественно истинен. Циклы в автоматной программе имеют другую природу по сравнению с циклами императивной программы. Не рекомендуется смешивать разные стили программирования, в частности использовать циклы типа **while** для конструирования автоматной программы.

Специализация сегмента кода на базе некоторого условия e(x) является популярной оптимизирующей трансформацией. Сегмент кода H: S заменяется на

```
H1: inv e(x); S
H2: inv ¬e(x); S
```

Далее реализуются упрощающие преобразования двух разных вхождений S на базе условий e(x) и ¬e(x).

Сегмент кода в виде суперпозиции B(s: s′); C, где s — переменные состояния автоматной программы, B(s: s′) — вызов программы-функции и C — программа-процесс, трансформируется в *редукцию суперпозиции*:

```
H: B(s: s′) #K
K: inv e(s); C
```

Применение редукции суперпозиции целесообразно в случае последующей специализации процесса C.

Передатчик в протоколе AAL-2

Метод трансформации требований для построения автоматной программы иллюстрируется на примере программы Передатчик в протоколе AAL-2, описанной на языке спецификаций SDL [14] в разд. 10.1 стандарта [9]. Сначала строится простейшая формализация требований, которая далее последовательно трансформируется в целях получения эффективной программы.

Уровень адаптации 2 (*AAL type 2*) в асинхронном способе передачи данных ATM (*Asynchronous Transfer Mode*) применяется для эффективной передачи низкоскоростных, коротких пакетов переменной длины в приложениях, чувствительных к задержкам. Используются три уровня передачи данных: подуровень конвергенции SSCS (*Service Specific Convergence Sublayer*), общий подуровень CPS (*Common Part Sublayer*) и уровень ATM.

Содержательное описание. Передатчик получает *пакеты* данных с уровня SSCS и плотно пакует их в последовательности *блоков данных*, посылаемых на уровень ATM. Очередной пакет поступает через сообщение CPSpacket(pd), где pd — *данные пакета* переменной длины. Передатчик строит *заголовок пакета* ph (*packet header*) длиной 3 октета[1]. *Пакет*, состоящий из ph и pd, плотно пакуется в один или несколько *блоков данных* (*protocol data units*). Блок данных состоит из *начального поля* STF (*start field*) длиной в 1 октет и *основной части* (*payload*) длиной 47 октетов. Пакет, не вместившийся в очередном блоке, продолжается в следующем (рис. 2). Начальное поле STF блока данных содержит длину в октетах перенесенной части пакета из предыдущего блока. Очередной построенный блок передается на уровень ATM посылкой сообщения ATM _ data("блок").

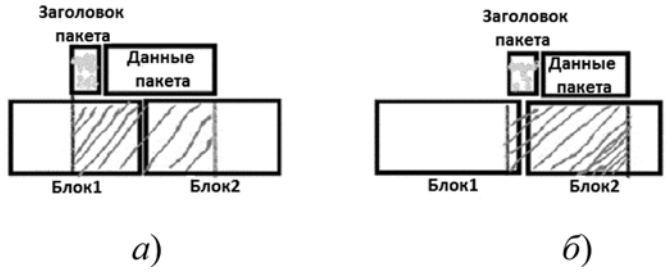


Рис. 2. Схема переноса пакета:
а — перенос данных; *б* — перенос заголовка

Очередной заполненный блок может быть отправлен на уровень ATM только после получения запроса на посылку — сообщения SEND _ request(). Построение очередного блока управляется таймером: по истечении времени Twait построение очередного блока завершается заполнением нулями оставшейся части блока.

Определим непосредственную формализацию представленных требований. Введем буфер packs для хранения пакетов, полученных через сообщения CPSpacket и еще не отправленных на уровень ATM. Когда длина буфера packs достигает или превышает 47 (длины информационной части блока), очередной сформированный блок посылается на уровень ATM сообщением ATM _ data при дополнительном условии — получении сообщения SEND _ request.

Окружение:

```
message CPSpacket(pd), SEND _ request,
ATM _ data(block).
```

Пакет и блок данных имеют тип DATA — список октетов:

```
type OCTET = byte;
type DATA = list(OCTET);
```

Использование списков вместо массивов упрощает программу. Константа nil обозначает пустой список; len(x) — длину списка x; x + y — кон-

[1] Октет состоит из 8 битов; термин «байт» не используется, так как имеет иной смысл.

катенацию списков x и y. При трансляции списки кодируются вырезками массивов [5, 6].

Состояние:

```
DATA packs = nil, OCTET stf =
= ConstructSTF(0), time t.
```

Локальные переменные:

```
DATA pd, block.
```

Локальные программы:

```
pred extract _ block(DATA packs: DATA block, packs')
  pre len(packs) ⩾ 47
  post packs = block + packs' & len(block) = 47;
```

Программа `extract _ block` выделяет очередной блок `block` из буфера `packs` при условии, что его длина не меньше 47. Результат программы `packs'` — оставшаяся часть буфера.

Программа `ConstructCPS _ PacketHeader(pd)` строит заголовок `ph` для пакета `pd`. Программа `ConstructSTF(x)` строит начальное поле `STF` для следующего блока, где `x` — длина перенесенной части предыдущего пакета. Программа `fill(packs)` дополняет буфер `packs` нулями до длины в 47 октетов.

Требования:

```
len(packs) < 47, CPSpacket(pd) →
  set t,
  packs' = packs + ConstructCPS _ PacketHeader(pd) + pd;
len(packs) ⩾ 47, SEND _ request() →
  extract _ block(packs: block, packs'),
  ATM _ data(stf + block),                                    (1)
  if(packs'≠nil) set t,
  stf' = ConstructSTF(min(len(packs'), 47));
0 < len(packs) < 47, t > Twait, SEND _ request() →
  ATM _ data(stf + fill(packs)),
  packs' = nil,
  stf' = ConstructSTF(0).
```

Возможна эффективная реализация требований: буфер `packs` "плывет" внутри достаточно длинного массива, а при приближении к его концу переписывается в начало.

Наша цель — используя требования (1), построить эффективную программу, соответствующую релизам [1, 9], в которых вместо `packs` используется буфер `buf`, вмещающий не более 47 октетов. Буфер `buf`, заголовок очередного пакета `ph` и сам пакет `pd` в сумме составляют буфер `packs`.

Трансформация 1. Удалим оператор присваивания `packs'` в первом правиле. Заменим `packs` на `buf + ph + pd` в требованиях (1). Соответствующая модификация требований представлена ниже.

Состояние:

```
DATA buf = nil, ph = nil, pd = nil, time t,
OCTET stf = ConstructSTF(0)
```

Локальные программы. Меняется программа `extract _ block`, реализующая перепись на буфер `buf` заголовка `ph` и пакета `pd`, полностью или частично.

```
pred extract _ block(DATA buf, ph, pd: DATA buf', ph', pd')
  pre buf + ph + pd ⩾ 47
  post buf + ph + pd = buf' + ph' + pd' & len(buf') = 47;
```

В роли очередного блока выступает полностью заполненный буфер `buf'`. Другие результаты — модифицированные заголовок `ph'` и пакет `pd'`.

Требования:

```
len(buf + ph + pd) < 47, CPSpacket(pd') →
  set t, buf' = buf + ph + pd,
  ph' = ConstructCPS _ PacketHeader(pd');
0 < len(buf + ph + pd) < 47, t > Twait, SEND _ request() →
  ATM _ data(stf + fill(buf + ph + pd)),
  buf' = nil, ph' = nil, pd' = nil,
  stf' = ConstructSTF(0);                                     (2)
len(buf + ph + pd) ⩾ 47, SEND _ request() →
  extract _ block(buf, ph, pd: buf', ph', pd'),
  ATM _ data(stf + buf'),
  if (ph' + pd' ≠ nil) set t,
  buf'' = nil;
  stf' = ConstructSTF(min(len(ph' + pd'), 47))
```

Для упрощения требований (2) предварительно следует провести перепись заголовка ph и пакета pd на буфер buf, как это реализуется программой extract _ block. В связи с этим новая версия требований получается применением трансформации редукции суперпозиции.

Единственное управляющее состояние в требованиях (2) обозначим через H. Ведем новое управляющее состояние K с инвариантом:

```
(len(buf + ph + pd) < 47 ⇒ ph = nil & pd = nil) &
(len(buf + ph + pd) ⩾ 47 ⇒ len(buf) = 47).
```

Инвариант истинен после переписи ph и pd на буфер buf.

В управляющем состоянии H реализуется перепись ph и pd на буфер buf с переходом на управляющее состояние K, в котором реализуются аналоги правил требований (2).

Локальные программы.

```
pred move(DATA buf, x: DATA buf′, x′)
  pre len(buf) ⩽ 47
  post buf′ + x′ = buf + x &
    (len(buf + x) ⩾ 47? len(buf′) = 47: x′ = nil);
```

Программа move переписывает список x на буфер buf. Результат x′ — остаток от переписи, возможно пустой. Программа работает и при заполненном буфере с результатом x′ = x.

Требования: (3)

```
H → move(buf, ph: buf′, ph′), move(buf′, pd: buf′′, pd′), K;                (3.1)
K, len(buf) < 47, CPSpacket(pd) →                                          (3.2)
   set t, ph = ConstructCPS _ PacketHeader(pd), H;
K, 0 < len(buf) < 47, t > Twait, SEND _ request() →                        (3.3)
   ATM _ data(stf + fill(buf)), buf′ = nil, ph′ = nil, pd′ = nil,
   stf′ = ConstructSTF(0), H;
K, len(buf) = 47, SEND _ request() →                                       (3.4)
   ATM _ data(stf + buf), if (ph + pd ≠ nil) set t,
   buf′ = nil; stf′ = ConstructSTF(min(len(ph + pd), 47)), H.
```

Неэффективность версии (3) в том, что программа move запускается для пустых ph и pd, а также в случае len(buf) = 47. Для устранения этих недостатков определим программу move в виде гиперфункции с тремя ветвями:

```
hyper move(DATA buf, x: DATA buf′ #1: DATA buf′ #2:DATA buf′, x′ #3)
pre len(buf) ⩽ 47 // общее предусловие
pre 1: len(buf + x) < 47
pre 2: len(buf + x) = 47
pre 3: len(buf + x) > 47
post 1: len(buf′) < 47 & buf′ = buf + x
post 2: len(buf′) = 47 & buf′ = buf + x
post 3: len(buf′) = 47 & buf + x = buf′ + x′ & x′≠nil;
```

Для первой и второй ветвей список x полностью переписан на буфер buf, причем для второй ветви буфер полностью заполнен. В третьей ветви после переписи остается непустой остаток x′.

Трансформация 2. В требовании (3.1) заменим вызовы программы move на эквивалентные вызовы гиперфункций. С этой целью определим выходы для всех трех ветвей в одну точку непосредственно после вызова гиперфункции. Например:

```
move(buf, ph: buf′ #G: buf′ #G: buf′, ph′ #G); G:
```

Трансформация 3. Введем новые управляющие состояния с инвариантами:

```
PART: inv len(buf) < 47 & ~ph & ~pd;
B1: inv len(buf) = 47 & ~ph & ~pd;
                  // ph и pd полностью переписаны
B2: inv len(buf) = 47 & ~ph; // ph полностью переписан
B3: inv len(buf) = 47 & ph′≠nil; // ph переписан частично
```

Здесь ~ph и ~pd обозначают условия ph = nil и pd = nil. Обнуление переменных ph и pd можно будет опустить, поскольку их значения далее использоваться не будут.

Трансформация 4. Проведем специализацию требования (3.4) для управляющих состояний B1, B2 и B3, а также требований (3.2) и (3.3) для управляющего состояния PART.

Трансформация 5. Проведем уточнение итоговых управляющих состояний во всех требованиях, в том числе определяемых выходами вызовов гиперфункций. С этой целью состояние в конце каждого требования сопоставляется с инвариантами управляющих состояний. При этом пропускаются "холостые" срабатывания правил.

Требования:

```
PART, CPSpacket(pd) →
   set t, ph = ConstructCPS _ PacketHeader(pd), H;
H → move(buf, ph: buf' #G: buf' #B2: buf', ph' #B3);
G → move(buf, pd: buf' #PART: buf' #B1: buf', pd' #B2);
PART, buf≠nil, t > Twait, SEND _ request() →
   ATM _ data(stf + fill(buf)),
   buf' = nil, stf' = ConstructSTF(0), PART;
B1, SEND _ request() → ATM _ data(stf + buf),                    (4)
   buf' = nil; stf' = ConstructSTF(0), PART;
B2, SEND _ request() → ATM _ data(stf + buf); set t,
   buf' = nil; stf' = ConstructSTF(min(len(pd), 47)), G;
B3, SEND _ request() → ATM _ data(stf + buf), set t,
   buf' = ph; stf' = ConstructSTF(min(len(ph + pd), 47)), G.
```

Требования (4) непосредственно переписываются в программу.

Программа:

```
process transfer() {
PART:  inv len(buf) < 47 & ~ph & ~pd;
    if (CPSpacket(pd)) {
     set t;
     ph = ConstructCPS _ PacketHeader(pd);
     move(buf, ph: buf' #G: buf' #B2: buf', ph' #B3)
G:     inv len(buf) < 47 & ~ph;
     move(buf, pd: buf #PART: buf' #B1: buf', pd' #B2)
    }else if (buf≠nil & t > Twait & SEND _ request()) {
     send ATM _ data(stf + fill(buf));
     buf = nil; stf = ConstructSTF(0) #PART
    }
B1:   inv len(buf) = 47 & ~ph & ~pd;
     receive SEND _ request();
     send ATM _ data(stf + buf); stf' = ConstructSTF(0);
     buf = nil #PART
B2:   inv len(buf) = 47 & ~ph;
     receive SEND _ request();
     send ATM _ data(stf + buf); set t;
     buf = nil; stf' = ConstructSTF(min(len(pd), 47)) #G
B3:   inv len(buf) = 47 & ph≠nil;
     receive SEND _ request();
     send ATM _ data(stf + buf); set t;
     buf = ph; stf' = ConstructSTF(min(len(ph + pd), 47));
     #G
}
```

Заключение

В настоящей работе рассмотрена проблема построения сложных автоматных программ на примере программы `Передатчик` в протоколе AAL-2. Ее описание в стандарте [9, разд. 10.1] — сложное и громоздкое, с плохой эргономикой. Использование аппарата гиперфункций и списков вместо массивов существенно снижает сложность программы `Передатчик` в нашем релизе [1]. Однако и эта программа сложна, потому что она воссоздает неудачный дизайн стандарта [9], наследуя к тому же дефекты в эффективности.

Построение простой и эффективной программы `Передатчик` начинается с более простой задачи, представленной требованиями (1). Далее применяется последовательность трансформаций, приводящая к требованиям (4). Построенная по ним эффективная программа проще аналогичной программы из работы [1] — она почти вдвое короче и содержит 5 управляющих состояний вместо 11 в работе [1].

В принципе можно было бы начать разработку программы с требований (3) и, проведя их специализацию, получить требования (4). Преимущество метода трансформации требований от простых к более сложным заключается в дополнительной возможности нахождения ошибок в требованиях. Каждая новая версия требований после очередной трансформации подвергается осмыслению и анализу, при этом достаточно часто ошибочные ситуации становятся более явными и заметными. В процессе трансформаций в исходных требованиях (1) было найдено семь ошибок. Отметим, что при разработке информационных систем обычно допускается много ошибок в требованиях, они трудно ищутся в процессе моделирования и тестирования и дорого обходятся.

Представленный метод построения автоматных программ применением трансформаций может быть использован лишь для сложных программ. Для построения большинства автоматных программ, в частности, для программ, представленных в работе [6], трансформации не нужны. В данной работе трансформации лишь частично классифицированы. Формализация и детальная классификация — цель дальнейших исследований. Применение трансформаций, в частности, планируется при построении программ для серии протоколов взаимного исключения.

Автор благодарен А. А. Шалыто за его работы по автоматному программированию, стимулировавшие исследования автора.

Работа выполнена при поддержке РФФИ, грант № 12-01-00686.

Список литературы

1. **Шелехов В. И.** Язык и технология автоматного программирования // Программная инженерия. 2014. № 4. С. 3—15. URL: http://persons.iis.nsk.su/files/persons/pages/automatProg.pdf

2. **Карнаухов Н. С., Першин Д. Ю., Шелехов В. И.** Язык предикатного программирования Р. Препринт № 153. Новосибирск: ИСИ СО РАН, 2010. 42 с. URL: http://persons.iis.nsk.su/files/persons/pages/plang12.pdf

3. **Shelekhov V. I.** Verification and Synthesis of Addition Programs under the Rules of Correctness of Statements // Automatic Control and Computer Sciences. 2011. Vol. 45, N 7. P. 421—427.

4. **Шелехов В. И.** Верификация и синтез эффективных программ стандартных функций в технологии предикатного программирования // Программная инженерия. 2011. № 2. С. 14—21.

5. **Шелехов В. И.** Разработка и верификация алгоритмов пирамидальной сортировки в технологии предикатного программирования. Препринт № 164. Новосибирск: ИСИ СО РАН, 2012. 30 с.

6. **Шелехов В. И.** Разработка автоматных программ на базе определения требований // Системная Информатика. 2015. № 1. С. 1—29. URL: http://persons.iis.nsk.su/files/persons/pages/req_tech.pdf

7. **Cockburn A.** Writing Effective Use Cases. Addison-Wesley, 2001. 270 p.

8. **Klahr D., Langley P., Neches R.** Production System Models of Learning and Development. Cambridge, Mass.: The MIT Press, 1987. 467 p.

9. **ITU-T** Recommendation I.363.2 (11/2000):"B-ISDN ATM Adaptation Layer Specification: Type 2 AAL". URL: http://men.axenet.ru/itu/ORIGINAL/I/T-REC-I.363.2-200011-I!!PDF-E.pdf

10. **IEEE** Recommended Practice for Software Requirements Specifications. Revision: 29/Dec/11.

11. **Mich L., Franch M., Novi Inverardi P.** Market research for requirements analysis using linguistic tools // Requirements Engineering. 2004. Vol. 9, N 1. P. 40—56.

12. **Leveson N., Heimdahl M., Hildreth H., Damon J.** Requirements Specification for Process-Control Systems // IEEE Transactions on Software Engineering. 1994. Vol. 20, N 4. P. 684—707.

13. **Zhang W., Beaubouef T., Statechart Ye H.** A Visual Language for Software Requirement Specification // International Journal of Machine Learning and Computing. 2012. Vol. 2, N 1. P. 52—61.

14. **Specification** and description language (SDL). ITU-T Recommendation Z.100 (03/93). URL: http://www.itu.int/ITU-T/studygroups/com17/languages/Z100.pdf

15. **Glinz M.** Problems and Deficiencies of UML as a Requirements Specification Language // 10th International Workshop on Software Specification and Design. 2000. P. 11—22.

16. **Bellini P., Mattolini R., Nesi P.** Temporal logics for real-time system specification // ACM Comp. Surveys. 2000. Vol. 32, N 1. P. 12—42.

17. **Arcaini P., Gargantini A., Riccobene E.** Online Testing of LTL Properties for Java Code // Hardware and Software: Verification and Testing. LNCS 8244. 2013. P. 95—111.

18. **Sunha A., Sharad M.** Modeling Firmware as Service Functions and Its Application to Test Generation // Hardware and Software: Verification and Testing. LCNS 8244. 2013. P. 61—77.

19. **Шпаков В. М**. Формализация и использование знаний о развитии процессов // Тр. 16-й Междунар. конф. "Проблемы управления и моделирования в сложных системах". Самара: Самарский научный центр РАН, 2014. С. 290—295.

20. **Brandt J., Gemünde M., Schneider K.** et al. Representation of synchronous, asynchronous, and polychronous components by clocked guarded actions // Design Automation for Embedded Systems. Springer, 2012. P. 1—35.

21. **Arvind H.** Bluespec: a language for hardware design, simulation, synthesis and verification // MEMOCODE'03 Proceedings of the First ACM and IEEE International Conference on Formal Methods and Models for Co-Design. 2003. P. 249—254.

22. **Brescani P., Perini A., Giorgini P.** et al. Modelling Early Requirements in Tropos: A Transformation-Based Approach // Agent-Oriented Software Engineering II. LNCS 2222. 2002. P. 151—168.

23. **Тумуров Э. Г., Шелехов В. И.** Определение требований к системе управления полетом квадрокоптера // Тр. 16-й Междунар. конф. "Проблемы управления и моделирования в сложных системах". Самара: Самарский научный центр РАН, 2014. С. 627—633.

24. **Alur R., Dill D. L.** A theory of timed automata // Theor. Comput. Sci. 1994. N 126. P. 183—235.

25. **Cooke D. E., Rushton J. N.** Taking Parnas's Principles to the Next Level: Declarative Language Design // Computer. 2009. Vol. 42, N 9. P. 56—63.

26. **Шелехов В. И.** Предикатное программирование. Учеб. пособие. Новосибирск: НГУ, 2009. 109 с.

V. I. Shelekhov, Head of Laboratory, A. P. Ershov Institute of Informatics Systems, Novosibirsk, Russia

Automata-Based Program Optimization by Applying Requirement Transformations

Automata-based software engineering includes methods to develop simple, reliable and effective programs for reactive systems. A program is constructed as a finite state automation presented in the form of control state hypergraph. A production system language for describing the functional requirements is proposed as the specification language for automata-based programs. For development of an effective automata-based program presented by the production set, the optimizing requirement transformations are used. These methods are illustrated on the ATM adaptation layer (type 2) protocol. The presented version of Sender program for the AAL2 protocol is essentially simple and efficient in comparison with the same program published in the AAL2 standard.

The optimizing transformations are the followings: program specialization, superposition reduction, equivalence replacement, and simplifications. The transformations applied are not automatic. They are included in the user's program development process. For large switching programs, an essential optimization effect is obtained by replacing programs to equivalent hyperfunctions.

Keywords: *program comprehension, automata-based programming, requirement specification, production system model, ATM Adaptation Layer*

References

1. **Shelekhov V. I.** Yazyk i tekhnologiya avtomatnogo programmirovaniya. *Programmnaya Ingeneria,* 2014, no. 4, pp. 3—15 (in Russian).

2. **Kharnaukhov N. S., Perchine D. Ju., Shelekhov V. I.** Yazyk predikatnogo programmirovaniya P. *Preprint no. 153.* Novosibirsk, ISI SB RAN, 2010, 42 p. (in Russian).

3. **Shelekhov V. I.** Verification and Synthesis of Addition Programs under the Rules of Correctness of Statements. *Automatic Control and Computer Sciences*, 2011, vol. 45, no. 7, pp. 421—427.

4. **Shelekhov V. I.** Verifikatsiya i sintez effektivnykh programm standartnykh funktsiy v tekhnologii predikatnogo programmirovaniya. *Programmnaya Ingeneriya*, 2011, no. 2, pp. 14—21 (in Russian).

5. **Shelekhov V. I.** Razrabotka i verifikatsiya algoritmov piramidalnoy sortirovki v tekhnologii predikatnogo programmirovaniya. *Preprint no. 164.* Novosibirsk, ISI SB RAN, 2012, 30 p. (in Russian).

6. **Shelekhov V. I.** Razrabotka avtomatnykh programm na baze opredeleniya trebovaniy. *Sistemnaya Informatika*, 2015, no. 1. pp. 1—29 (in Russian), available at: http://persons.iis.nsk.su/files/persons/pages/req_tech.pdf

7. **Cockburn A.** *Writing Effective Use Cases.* Addison-Wesley, 2001, 270 p.

8. **Klahr D., Langley P., Neches R.** *Production System Models of Learning and Development.* Cambridge, Mass., The MIT Press, 1987, 467 p.

9. **ITU-T** Recommendation I.363.2 (11/2000): *"B-ISDN ATM Adaptation Layer Specification*: *Type 2 AAL"*, available at:http://men.axenet.ru/itu/ORIGINAL/I/T-REC-I.363.2-200011-I!!PDF-E.pdf

10. **IEEE** Recommended Practice for Software Requirements Specifications. Revision: 29/Dec/11.

11. **Mich L., Franch M., Novi Inverardi P.** Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 2004, vol. 9, no. 1, pp. 40—56.

12. **Leveson N., Heimdahl M., Hildreth H., Damon J.** Requirements Specification for Process-Control Systems. *IEEE Transactions on Software Engineering*, 1994, vol. 20, no. 4, pp. 684—707.

13. **Zhang W., Beaubouef T., Statechart Ye. H.** A Visual Language for Software Requirement Specification. *International Journal of Machine Learning and Computing*, 2012, vol. 2, no. 1, pp. 52—61.

14. **Specification** and description language (SDL). *ITU-T Recommendation Z.100* (03/93), available at: http://www.itu.int/ITU-T/studygroups/com17/languages/Z100.pdf

15. **Glinz M.** Problems and Deficiencies of UML as a Requirements Specification Language. *10th International Workshop on Software Specification and Design*, 2000, pp. 11—22.

16. **Bellini P., Mattolini R., Nesi P.** Temporal logics for real-time system specification. *ACM Comp. Surveys*, 2000, vol. 32, no. 1, pp. 12—42.

17. **Arcaini P., Gargantini A., Riccobene E.** Online Testing of LTL Properties for Java Code. *Hardware and Software: Verification and Testing.* LNCS 8244, 2013, pp. 95—111.

18. **Sunha A., Sharad M.** Modeling Firmware as Service Functions and Its Application to Test Generation. *Hardware and Software: Verification and Testing.* LCNS 8244, 2013, pp. 61—77.

19. **Shpakhov V. M.** Formalizatsiya i ispolzovaniye znaniy o razvitii protsessov. *Tr. 16-y mezhd. konf. "Problemy upravleniya i modelirovaniya v slozhnykh sistemakh"* (16-th International Conference "Problems of control and modeling in complex systems"). Samara, Samara Scientific centre RAN, 2014, pp. 290—295 (in Russian).

20. **Brandt J., Gemunde M., Schneider K., Shukla S. K., Talpin J.-P.** Representation of synchronous, asynchronous, and polychronous components by clocked guarded actions. *Design Automation for Embedded Systems.* Springer, 2012, pp. 1—35.

21. **Arvind H.** Bluespec: a language for hardware design, simulation, synthesis and verification. *MEMOCODE'03 Proceedings of the First ACM and IEEE International Conference on Formal Methods and Models for Co-Design*, 2003, pp. 249—254.

22. **Brescani P., Perini A., Giorgini P., Giunchiglia F., Mylopoulos J.** Modelling Early Requirements in Tropos: A Transformation-Based Approach. *Agent-Oriented Software Engineering II.* LNCS 2222, 2002, pp. 151—168.

23. **Tumurov E. G., Shelekhov V. I.** Opredeleniye trebovaniy k sisteme upravleniya poletom kvadrokoptera (Requirement specification of quad_rotor flight control system). *Tr. 16-y Mezhdunar. konf. "Problemy upravleniya i modelirovaniya v slozhnykh sistemakh"*(16-th International Conference "Problems of control and modeling in complex systems"). Samara, Samara Scientific centre RAN, 2014, pp. 627—633 (in Russian).

24. **Alur R., Dill D. L.** A theory of timed automata. *Theor. Comput. Sci*, 1994, no. 126, pp. 183—235.

25. **Cooke D. E., Rushton J. N.** Taking Parnas's Principles to the Next Level: Declarative Language Design. *Computer*, 2009, vol. 42, no. 9, pp. 56—63.

26. **Shelekhov V. I.** *Predikatnoye programmirovaniye*. Ucheb. posobiye (Predicate programming. Tutorial). Novosibirsk, NSU, 2009, 109 p. (in Russian).

УДК 004.925.83

И. Е. Ландовская, аспирант, e-mail: nairy@rambler.ru, **В. Д. Фроловский,** д-р техн. наук, проф., e-mail: frolovskij@corp.nstu.ru, **В. В. Ландовский,** канд. техн. наук, доц., e-mail: landovskij@corp.nstu.ru, Новосибирский государственный технический университет

Моделирование деформации тканых материалов с использованием методов параллельной обработки данных

Рассмотрено моделирование поведения тканых материалов с учетом их деформационных свойств методом частиц, где для решения системы дифференциальных уравнений используется схема с перешагиванием. Приведено описание разработанных авторами алгоритмов параллельной обработки данных для осуществления вычислений на центральном процессоре как в системах с общей памятью, так и в системах с распределенной памятью с учетом особенностей задачи. Представлен метод реализации параллельных вычислений на графическом процессоре. Приведены результаты, полученные в ходе проведения численных экспериментов для описанных алгоритмов параллельной обработки данных.

Ключевые слова: *моделирование тканых материалов, методы параллельной обработки данных, деформационные свойства ткани, метод частиц, схема с перешагиванием*

Введение

Задача моделирования деформации ткани является одной из наиболее интересных для исследования и сложных задач компьютерной графики. Как будет выглядеть ткань с определенными свойствами на определенном объекте? Как с изменением свойств ткани изменяется ее драпировка? Ответы на эти и другие вопросы в наглядной форме и должно давать компьютерное моделирование ткани. Важным фактором в постановке этой задачи является не только достижение наибольшей визуальной реалистичности, но, возможно в большей степени, обеспечение соответствия модели физическим характеристикам ткани, соответствия моделируемых деформаций реальным.

Уже не одно столетие исследователи разных стран пытаются создать модель тканого материала, которая будет отражать все свойства реального полотна. Одна из первых попыток создания модели ткани была предпринята еще в конце XIX века русским математиком и механиком П. Чебышевым. С того времени было предложено большое число методов (подходов) деформационного моделирования поверхностей тканей в пространстве, которые в основном можно разделить на три группы: физические, геометрические и гибридные.

В физических методах ткань, как правило, представляет собой систему взаимосвязанных частиц, движение которых описывается законами механики, а процесс моделирования сводится к решению системы дифференциальных уравнений с начальными условиями и нахождению траекторий движения частиц. Одним из первых использование такого подхода предложил D. Terzopoulos [1].

Модели на основе геометрических методов рассматривают не физические свойства ткани, а сгибы и складки, которые представляются в виде геометрических уравнений. Одной из первых работ, посвященных данной тематике, является работа J. Weil, опубликованная в 1986 г. [2].

Первым, кто предложил использовать гибридный метод, был I. J. Rudomin [3]. Он полагал, что можно ускорить процесс моделирования, если, получив некоторый результат с помощью геометрического подхода, затем использовать его как исходное положение динамической модели. Однако суммарные вычислительные затраты предложенного метода не оправдали надежд. Другим примером гибридного подхода является работа [4], где для определения формы ткани решалась задача поиска положения, соответствующего минимуму энергии, а затем в промежутках между частицами геометрическим методом моделировались складки.

Из всех описанных методов деформационного моделирования наиболее эффективными в плане повторения поведения реальных материалов считаются физические методы. Одним из успешно развивающихся подходов физического моделирования ткани является метод частиц [5]. Модели частиц при-

меняют для моделирования самых разных явлений, таких как водопады, взрывы, вихревые поля и т. д. Каждая частица имеет набор атрибутов, таких как масса, положение, скорость. Состояние физической системы определяется атрибутами конечного ансамбля частиц, а эволюция системы — законами взаимодействия этих частиц.

Очевидно, что чем большим числом частиц представлена модель ткани, тем точнее она воспроизводит деформационные свойства реального материала при компьютерном моделировании. Пределом детализации модели ткани является сетка с шагом, сравнимым с толщиной нити тканого полотна. При такой детализации значительно увеличивается число частиц модели, что, естественно, приводит к увеличению времени моделирования, которое возрастает пропорционально увеличению числа частиц, которыми представлена сетка тканого материала. Сокращения времени расчетов такой модели можно добиться, воспользовавшись средствами параллельной обработки данных. Такие средства помогают повысить скорость получения результатов за счет использования вычислительных ресурсов сразу нескольких ядер одного центрального процессора, нескольких центральных процессоров вычислительного кластера или сотен ядер графического процессора видеоадаптера одновременно.

Целью данной статьи является представление разработанных авторами алгоритмов параллельной обработки данных и программных реализаций для распараллеливания вычислений при компьютерном моделировании поведения тканых материалов и их взаимодействия с твердотельными многогранными объектами. При этом учитывались следующие особенности и варианты решения задачи:

— на центральном процессоре (для систем с общей памятью и распределенной памятью);

— на графическом процессоре видеоадаптера.

1. Модель ткани и ее деформационные свойства

Ткань представляет собой пространственную сетку, образованную переплетением в определенной последовательности двух взаимно перпендикулярных систем нитей. Нити, идущие вдоль полотна, называют основой; нити, расположенные поперек полотна, — утком [6]. Поэтому при компьютерном моделировании тканый материал рассматривают как систему взаимодействующих частиц (узлов), которые размещены в точках пересечения нитей основы и нитей утка [7]. На рис. 1 представлена дискретная модель ткани.

Одним из важнейших этапов построения модели является определение сил, которые характеризуют взаимодействия частиц. Далее по тексту для краткости изложения будем называть их взаимодействиями. Для определения основных взаимодействий частиц необходимо рассмотреть деформационные свойства тканого полотна.

Основные взаимодействия, которые происходят на уровне нити, это: растяжение—сжатие; изгиб и сдвиг [8]. Учитывая, что ткань достаточна легкая и масса ткани в удаленных узлах оказывает пренебрежимо малое влияние на каждую рассматриваемую частицу, предположим, что на каждую внутреннюю частицу влияют 12 соседних частиц. На рис. 2 взаимодействия растяжения—сжатия, сдвига и изгиба обозначены связями *1*, *2* и *3* соответственно.

Введем обозначения: $F_{s,\ ijkl}$ — сила взаимодействия растяжения—сжатия, с которой частица P_{kl} воздействует на связанную с ней частицу P_{ij}; $F_{t,\ ijkl}$, $F_{b,\ ijkl}$ — силы взаимодействий сдвига и изгиба соответственно.

Взаимодействие растяжения—сжатия имеет место, когда расстояние между двумя соседними частицами

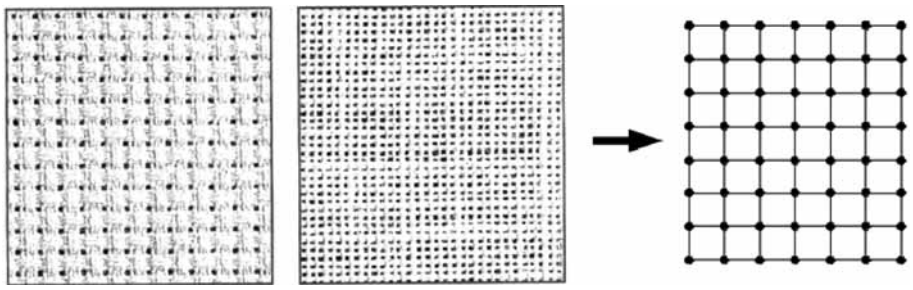


Рис. 1. Дискретная модель ткани

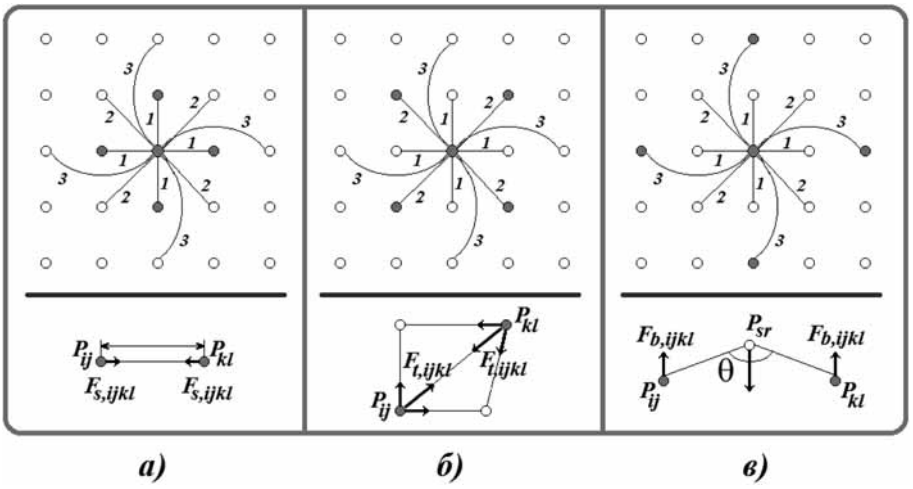


Рис. 2. Типы взаимодействий:
а — растяжение—сжатие, *б* — сдвиг, *в* — изгиб

(рис. 2, *а*, связи *1*) отличается от расстояния между ними в состоянии покоя. В случае растяжения—сжатия направление сил очевидно, так как во взаимодействии учитываются положения только двух частиц. Силу, возникающую при взаимодействиях растяжения—сжатия на n-м шаге интегрирования между соседними частицами P_{ij} и P_{kl}, $k = i \pm d$, $l = j \pm p$; $d, p \in \{0; 1\}$, $d \neq p$, можно получить, используя выражение

$$F_{s,ijkl} = \frac{F_{spr}(L)}{S_{pr}} S_{yas},$$

где $F_{spr}(L)$ — сила, возникающая в пробе материала, при растяжении его вдоль нити основы или утка на L процентов (значение силы определяется исходя из диаграммы "нагрузка—удлинение" материала (рис. 3) в соответствии с ГОСТ 3813—72 отдельно для нитей основы и нитей утка); S_{pr} — площадь пробы (размеры проб для шерстяных и полушерстяных тканей — 50×100 мм, для всех остальных — 50×200 мм); S_{yas} — площадь ячейки сети модели, которой представлен тканый материал.

Результирующая сила взаимодействия растяжения—сжатия для частицы P_{ij} определим как сумму

$$F_{s,ij} = \sum_{kl \in R_{s,ijkl}} F_{s,ijkl},$$

где $R_{s,\ ijkl}$ — множество индексов частиц, связанных с частицей P_{ij} взаимодействием растяжения—сжатия.

Взаимодействие сдвига (рис. 2, *б*, связи 2) можно выразить через взаимодействие растяжения—сжатия частиц, образующих диагональ ячейки. Силу, возникающую при взаимодействии сдвига на n-м шаге интегрирования между соседними частицами P_{ij} и P_{kl}, $k = i + d$, $l = j + p$; $d, p \in \{-1, 1\}$, можно получить, используя выражение

$$F_{t,ijkl} = \frac{F_{tpr}(L)}{S_{pr}} S_{yas},$$

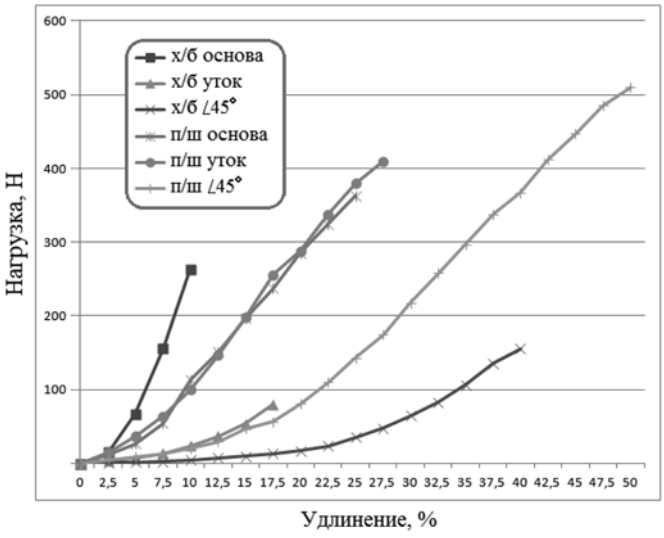


Рис. 3. Диаграмма "нагрузка—удлинение" для хлопчатобумажной (х/б) и полушерстяной (п/ш) тканей

где $F_{tpr}(L)$ — сила, возникающая в пробе материала при растяжении его под углом 45° к нити основы на L процентов (значение силы определяется исходя из диаграммы "нагрузка—удлинение" материала (см. рис. 3) в соответствии с ГОСТ 3813—72).

Результирующая сила взаимодействия сдвига для частицы P_{ij} определяется как сумма

$$F_{t,ij} = \sum_{kl \in R_{t,ijkl}} F_{t,ijkl},$$

где $R_{t,\ ijkl}$ — множество индексов частиц, связанных с частицей P_{ij} взаимодействием сдвига.

Последний вид взаимодействия — взаимодействие изгиба (см. рис. 2, *в*, связи *3*) можно выразить как функцию угла, образованного тремя последовательными частицами, лежащими на одной прямой (нити). Сила, возникающая при взаимодействии изгиба на n-м шаге интегрирования между частицами P_{ij} и P_{kl}, $k = i \pm d$, $l = j \pm p$; $d, p \in \{0, 2\}$, $d \neq p$, определяется исходя из выражения

$$F_{b,ijkl} = \frac{EI_{pr}}{b^2(1-\nu^2)N} \Delta\theta,$$

где EI_{pr} — жесткость пробы материала при изгибе; b — длина ребра ячейки сети; $\Delta\theta$ — угол между двумя ребрами (угол изгиба нити); ν — значение коэффициента Пуассона для данного материала; N — число ячеек сети модели ткани, которые поместятся по ширине пробы (размеры пробы составляют 160×30 мм).

Результирующую силу взаимодействия изгиба для частицы P_{ij} через частицу P_{sr} определим как сумму

$$F_{b,ij} = \sum_{kl \in R_{b,ijkl}} F_{b,ijkl} + \sum_{sr \in R_{b,ijsr}} \left(-F_{b,ijsr}\right),$$

где $R_{b,\ ijkl}$ — множество индексов частиц, связанных с частицей P_{ij} взаимодействием изгиба; $R_{b,\ ijsr}$ — множество индексов частиц, для которых частица P_{ij} является узлом, через который происходит изгиб.

В итоге получаем $F_{int,\ ij}$ — внутреннюю силу — суммарный результат всех внутренних сил, действующих на частицу:

$$F_{int,ij} = \sum_{kl \in R_{s,ijkl}} F_{s,ijkl} + \sum_{kl \in R_{t,ijkl}} F_{t,ijkl} + \\ + \sum_{kl \in R_{b,ijkl}} F_{b,ijkl} + \sum_{sr \in R_{b,ijsr}} \left(-F_{b,ijsr}\right),$$

где $R_{s,ijkl}$, $R_{t,\ ijkl}$, $R_{b,\ ijkl}$ — множество индексов частиц, связанных с частицей P_{ij} взаимодействиями растяжения—сжатия, сдвига и изгиба соответственно; $R_{b,\ ijsr}$ — множество индексов частиц, для которых частица P_{ij} является узлом, через который происходит изгиб.

Учитывая, что движение осуществляется в трехмерном пространстве, запишем выражения для проекций сил на оси координат

$$F_{ijkl,x} = F_{ijkl} \frac{\left(x_{kl} - x_{ij}\right)}{\sqrt{\left(x_{kl} - x_{ij}\right)^2 + \left(y_{kl} - y_{ij}\right)^2 + \left(z_{kl} - z_{ij}\right)^2}};$$

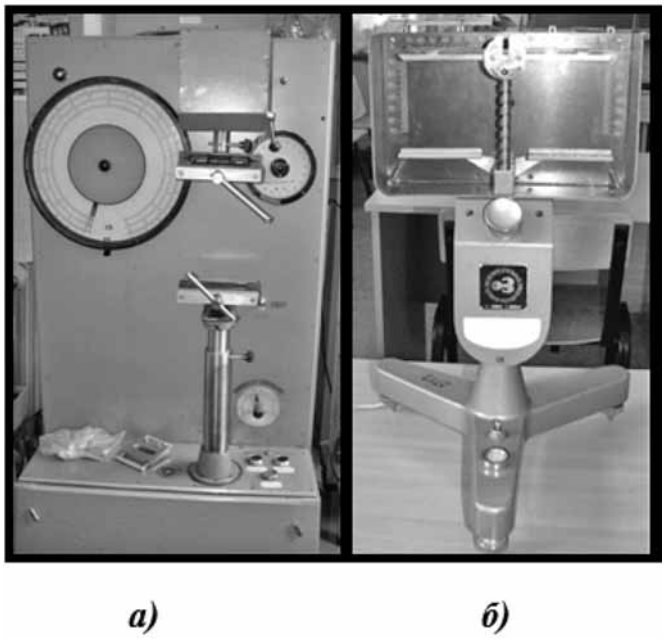
$$F_{ijkl,y} = F_{ijkl}\frac{\left(y_{kl} - y_{ij}\right)}{\sqrt{\left(x_{kl} - x_{ij}\right)^2 + \left(y_{kl} - y_{ij}\right)^2 + \left(z_{kl} - z_{ij}\right)^2}};$$

$$F_{ijkl,z} = F_{ijkl}\frac{\left(z_{kl} - z_{ij}\right)}{\sqrt{\left(x_{kl} - x_{ij}\right)^2 + \left(y_{kl} - y_{ij}\right)^2 + \left(z_{kl} - z_{ij}\right)^2}},$$

где F_{ijkl} — результирующая сила любого из видов перечисленных ранее взаимодействий; x_{ij}, y_{ij}, z_{ij} — координаты частицы P_{ij}, которая является началом вектора силы; x_{kl}, y_{k1}, z_{kl} — координаты частицы P_{kl}, которая является концом вектора силы.

Испытания материалов для получения характеристик деформации растяжения—сжатия и сдвига проводят на разрывной машине РТ-250 (рис. 4, *а*), а для получения характеристик изгиба — на приборе ПТ-2 (рис. 4, *б*) согласно ГОСТ 3813—72 и ГОСТ 10550—93 соответственно [6].

Следует отметить, что для получения реалистичных результатов сборки ткань должна быть представлена равномерной сеткой с ячейкой квадратной формы, чтобы действующие силы, возникающие в моделируемом изделии, были приближены по модулю и направлению к силам, возникающим в реальном материале при его деформациях. Кроме того, если готовое изделие должно состоять из нескольких частей (деталей) или на детали изделия предусмотрены выточки, то в местах соединений следует представить детали как совокупность контура и сетки ткани. При этом на контуре каждой детали изделия должны быть размещены частицы связи с другой деталью, отвечающие за соединение этих деталей в процессе сборки. Между частицами связей, принадлежащими



а) *б)*

Рис. 4. Измерительные приборы:
а — разрывная машина РТ-250; *б* — прибор ПТ-2

одной детали, в процессе моделирования будут возникать взаимодействия растяжения—сжатия и изгиба, а коэффициенты деформации для частиц связей определяют теми же методами, что и коэффициенты деформации самого материала.

2. Математическая модель ткани

Движение всей системы можно описать обобщенными перемещениями в трехмерном пространстве:

$$r_{ij}(t) = \left\{x_{ij}(t), y_{ij}(t), z_{ij}(t)\right\},$$

где $x_{ij}(t)$, $y_{ij}(t)$, $z_{ij}(t)$ — координаты частицы в трехмерном пространстве; t — время.

На каждом временном слое ищут положения узлов в пространстве. При этом каждая из частиц обладает некоторой массой, находится в гравитационном поле, взаимодействует с окружающей средой и соседними частицами [9]. Тогда уравнение движения частицы P_{ij} имеет следующий вид:

$$m_{ij}r''_{ij} = m_{ij}g - m_{ij}c_{ij}r'_{ij} + \sum_{kl \in R_{ijkl}} F_{int}(r_{ij}, r_{kl}),$$

где m_{ij} — масса частицы;c_{ij} — константа демпфирования; составляющая $\left(-m_{ij}c_{ij}r'_{ij}\right)$ представляет собой потери энергии, связанные с взаимодействием частицы с окружающей средой; g — ускорение свободного падения. Последняя составляющая уравнения движения представляет собой результирующую силу взаимодействий между частицами; R_{ijkl} — множество индексов узлов, связанных с узлом P_{ij}.

Для решения системы следует воспользоваться схемой с перешагиванием (*leapfrog scheme*) [5]. Этот выбор обусловлен тем обстоятельством, что для воспроизведения на компьютере реального поведения материала число частиц в моделируемом полотне должно быть велико, и к тому же каждое дополнительное вычисление силы требует больших временны́х затрат. Схема с перешагиванием является методом второго порядка точности и, в сравнении с классическим явным методом Эйлера, обладает большей устойчивостью, что позволяет на порядок увеличить шаг интегрирования:

$$\mathbf{V}_{n+1} = \mathbf{V}_n + h\mathbf{M}^{-1}\mathbf{F}(\mathbf{r}_n, \mathbf{V}_n),\ \mathbf{r}_{n+1} = \mathbf{r}_n + h\mathbf{V}_{n+1},$$

где $\mathbf{r}_n$ и $\mathbf{V}_n$ — векторы положений и скоростей частиц на n-м шаге интегрирования; h — шаг интегрирования; $\mathbf{F}(\mathbf{r}, \mathbf{V})$ — вектор-функция, описывающая действие внутренних и внешних сил на ткань, $\mathbf{M}$ — матрица инерции — диагональная матрица, описывающая распределение масс частиц ткани.

3. Общая схема моделирования и программная реализация

Общую схему моделирования процесса сборки изделий из тканых материалов на поверхности твердотельного многогранного объекта с учетом их де-

формационных свойств можно разделить на четыре основных этапа:

- построение сетки частиц деталей;
- создание соединений между деталями;
- расстановка объектов и деталей;
- моделирование процесса сборки изделия на поверхности твердотельного манекена.

На основе представленной схемы моделирования авторами данной статьи был разработан программный комплекс для тестирования работоспособности и эффективности предлагаемых методов и алгоритмов. Программный комплекс реализован в среде Microsoft Visual Studio 2012, на языке C++, с использованием графической библиотеки OpenGL и механизмов написания параллельных программ OpenMP, MPI и CUDA.

Общая схема работы программного комплекса моделирования поведения ткани с учетом входных и выходных данных представлена на рис. 5.

4. Алгоритмы параллельной обработки данных

Основным недостатком всех явных методов, в том числе и метода с перешагиванием, является то, что для обеспечения устойчивости необходимо выбирать достаточно малые значения шага, что приводит к значительному увеличению времени на получение результатов моделирования. Моделирование взаимодействия каждой частицы ткани с многогранным объектом, грани которого представляют собой ориентированные в пространстве треугольники, также является достаточно затратным по времени процессом. К тому же увеличение размерности модели заметно повышает реалистичность результатов, однако при этом существенно замедляет процесс моделирования. Так, например, после каждого шага интегрирования наступает этап, на котором определяется, взаимодействовала ли какая-либо частица ткани

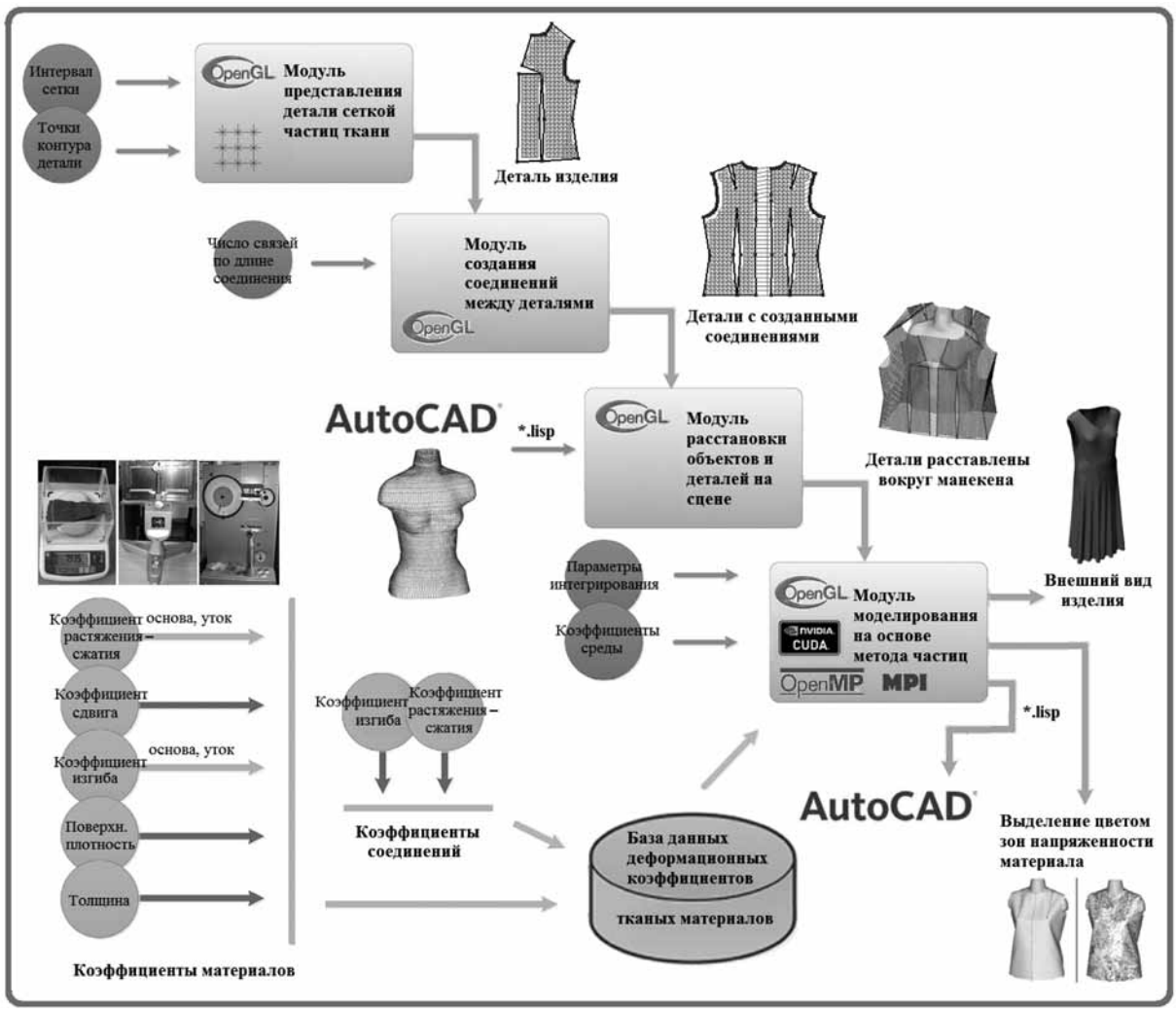


Рис. 5. Общая схема работы программного комплекса моделирования поведения ткани

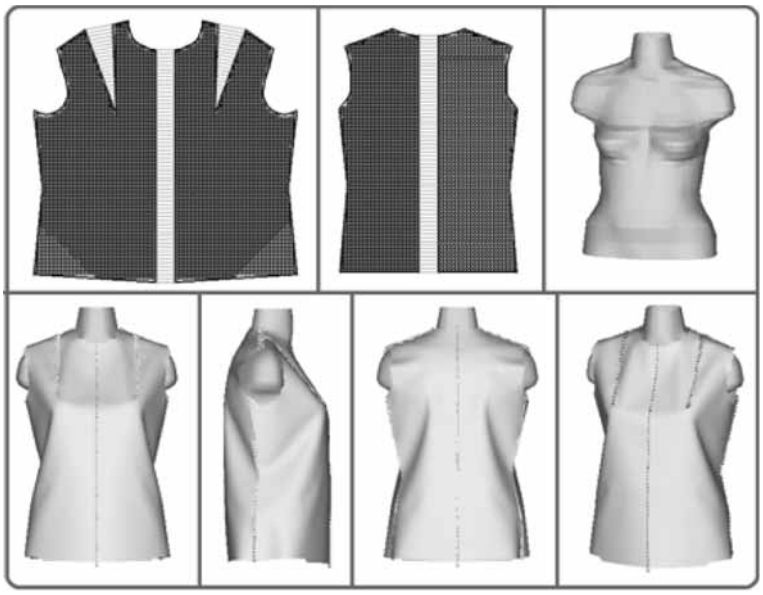


Рис. 6. Пример базовой конструкции, детали которой состоят из 6287 частиц, объекта, включающего 4800 граней, и готового изделия после сборки

с какой-нибудь гранью объекта. Если в некоторый момент времени частица пересекла грань объекта, то необходимо скорректировать ее положение и скорость. Таким образом, основной задачей на данном этапе является поиск столкновений, т. е. пересечений траекторий движения частиц ткани с гранями твердотельного объекта. Процедура проверки столкновений является достаточно ресурсоемкой задачей, ведь для объекта, представленного совокупностью M треугольных граней, и ткани, представленной сеткой из N частиц, на каждом шаге интегрирования проводится MN процедур проверки пересечения отрезка и треугольной грани в трехмерном пространстве. В итоге для объекта, включающего 4800 граней, и тканого материала, состоящего из 6287 частиц, следует провести более 30 млн операций проверки на каждом шаге интегрирования. Для сборки базовой конструкции в среднем требуется провести 500 000 шагов интегрирования, где шаг равен 0,00001 с. Пример базовой конструкции, детали которой состоят из 6287 частиц, и объекта, включающего 4800 граней, представлен на рис. 6.

4.1. Распараллеливание данных на центральном процессоре для систем с общей и распределенной памятью

Сокращения времени расчетов можно добиться, применяя методы параллельной обработки данных, которые смогут повысить скорость получения результатов за счет одновременного использования вычислительных ресурсов нескольких ядер одного центрального процессора или ресурсов нескольких центральных процессоров одного вычислительного кластера [10].

Наиболее оптимальным решением является разработка алгоритма параллельной обработки данных для осуществления расчетов на центральном процессоре, который подходит для систем как с общей, так и с распределенной памятью.

В процессе моделирования твердотельный многогранный объект, на поверхности которого происходит сборка, является неподвижным, а изменение координат происходит только у частиц ткани за счет действия на них различных сил, имитирующих воздействие окружающей среды. К их числу относятся стягивающая сила, сила тяжести, сопротивление воздуха, сила трения об объект и пр.

Так как для решения уравнений движения частиц ткани применяется явный метод, а именно схема с перешагиванием (см. разд. 2), то на каждом шаге интегрирования вначале вычисляется новое значение скорости, которое затем используется для вычисления значений координат на этом шаге. Этой особенностью можно воспользоваться при создании алгоритма параллельной обработки данных для систем как с общей, так и с распределенной памятью.

В то же время, как было описано в разд. 1, для вычисления скорости частицы на n-м шаге интегрирования нужно учитывать значения координат ее соседних частиц с $(n-1)$-го шага интегрирования. Следовательно, в алгоритме параллельной обработки данных для систем с распределенной памятью нужно предусмотреть этап своевременного обмена данными между процессами.

В контексте исследования, результаты которого представлены в настоящей статье, ткань рассматривается как система частиц, которые размещены в точках пересечения продольных и поперечных нитей. Исходными данными для построения сеточной модели детали изделия является контур этой детали (выкройки), который в общем случае представляет собой произвольный многоугольник (рис. 7).

Следовательно, в программе систему частиц, соответствующую сетке детали изделия, удобнее

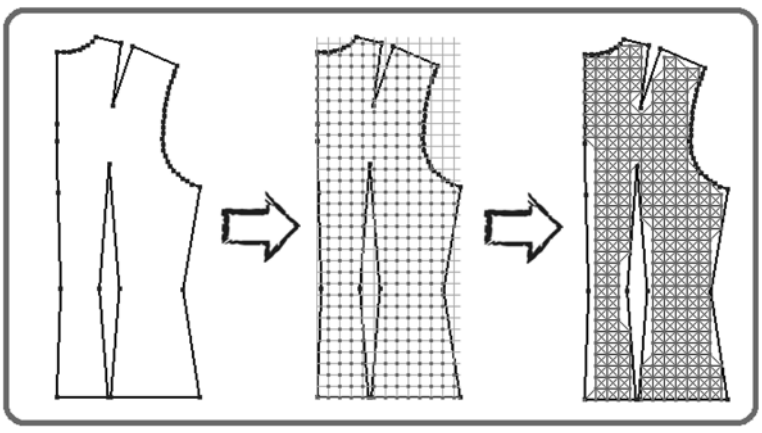


Рис. 7. Представление детали изделия сеткой частиц

всего представить в виде двумерного динамического массива. Число строк массива соответствует максимальному числу частиц по длине сетки детали, а число столбцов — максимальному числу частиц сетки детали по ее ширине. Если элементу массива не соответствует частица сетки детали, то он имеет значение *NULL*. При этом определить взаимодействующие частицы можно исходя из позиции элемента. Схематично двумерный массив для хранения данных о частицах сетки детали (далее для краткости — частицах детали) представлен на рис. 8.

При работе в системах с распределенной памятью передача данных между процессами всего массива координат частиц деталей, число которых составляет несколько тысяч, занимает достаточно большую часть от общего времени вычислений и сводит преимущества параллелизма к минимуму. Для уменьшения временных потерь на передачу данных между процессами и с учетом особенностей задачи моделирования был разработан усовершенствованный алгоритм. Он позволяет процессам обмениваться только четырьмя строками массива, что значительно повышает его эффективность. Этот алгоритм учитывает тот факт, что для вычисления значений координат точки ткани используются значения координат только 12 ее соседних частиц.

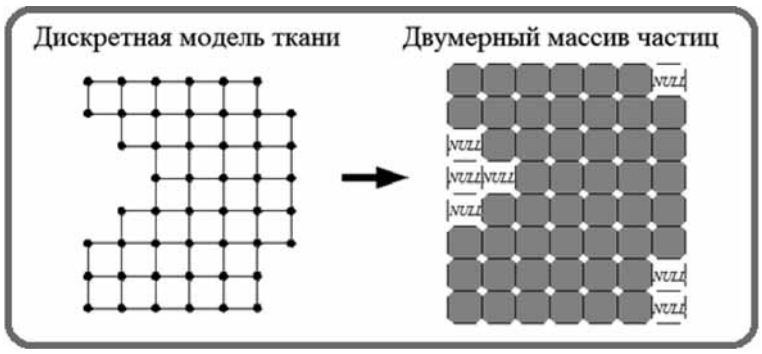


Рис. 8. Структура для хранения данных о частицах сетки детали

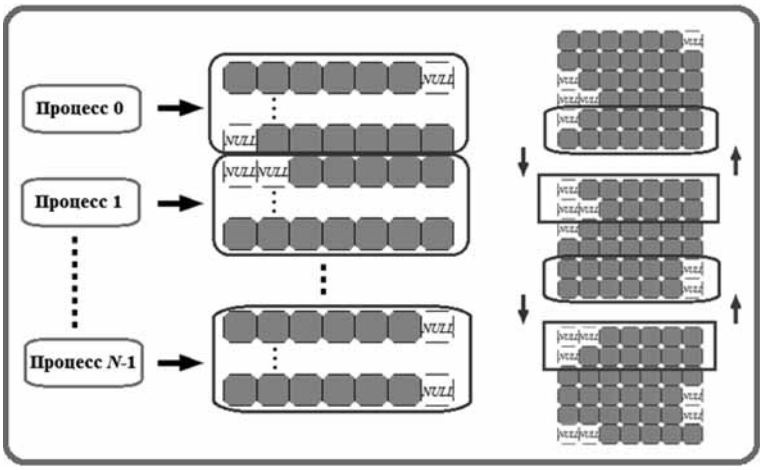


Рис. 9. Разделение данных и обмен между процессами

При моделировании двумерный массив, в котором хранятся данные о частицах детали, разделяется на число частей, равное числу запущенных процессов в программе [11]. Каждый процесс обрабатывает только свою часть массива, а затем происходит обмен данными, при котором процессы обмениваются между собой значениями координат двух крайних строк своей части массива (рис. 9).

В памяти каждого процесса создается отдельная копия данных массива частиц многогранного объекта так, что при моделировании процесс обрабатывает пересечения только своей части массива частиц детали с гранями объекта. Такой подход позволяет избежать лишнего обмена данными между процессами и приводит к резкому сокращению времени вычислений.

Параллелизм программы достигается как на уровне потоков, за счет применения библиотеки OpenMP, так и на уровне процессов — для этого применяется библиотека MPICH2. В каждом процессе порождается заданное число потоков, что полностью соответствует SPMD-модели, представленной на рис. 10. Данный алгоритм может быть использован в системах как с общей, так и с распределенной памятью, что делает его более универсальным.

Эффективность данного подхода к распараллеливанию данных для моделирования тканых материалов подтверждена численными экспериментами.

Часть алгоритма параллельной обработки данных, предназначенная для работы на многоядерном центральном процессоре в системах с общей памятью, была реализована в двух вариантах:

- порождение потоков OpenMP в начале моделирования и уничтожение их после осуществления всех шагов интегрирования;
- порождение потоков OpenMP на каждом шаге интегрирования и уничтожение их при окончании вычислений на данном шаге.

Тестирование проводилось на четырехъядерном процессоре Intel Core i7-4770K Haswell с частотой 3500 МГц и активированным гипертрейдингом. Данные, полученные в результате экспериментов, приведены в табл. 1.

Из данных, представленных в табл. 1, видно, что существенная разница во времени между применением двух разных способов порождения потоков возникает начиная с моделирования 10 000 000 итераций. Вместе с тем, как было отмечено ранее, для сборки изделия в среднем хватает 500 000 итераций. Таким образом, для решения задачи моделирования поведения тканых материалов нет различий в способе порождения потоков. Однако следует отметить, что разница во времени между применением двух представленных вариантов порождения потоков при вычислении более чем 10 000 000 итераций является тем большей в процентном соотношении к общему времени

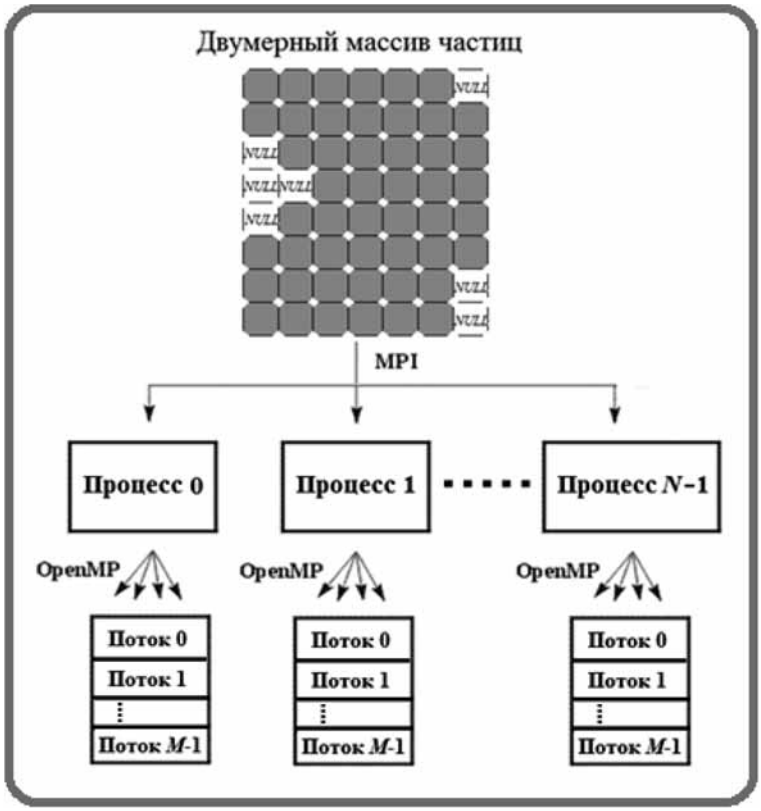


Рис. 10. Модель SPMD: одна программа, несколько потоков данных

моделирования, чем меньше точек содержат ткань и объект.

Представленные алгоритмы были также протестированы в восьмиядерной многопроцессорной среде, на кафедре вычислительной техники Новосибирского государственного технического университета с использованием вычислительного кластера с процессорами Intel Xeon E5310 Clovertown с частотой 1600 МГц и общей памятью 32 Гбайт. Тестирование проводилось с использованием различных контуров деталей и различных типов объектов. Все проведенные вычисления показывают приблизительно одинаковые коэффициенты сокращения временных затрат. Результаты одного из таких экспериментов представлены в табл. 2 и 3. Несмотря на то что тестирование алгоритмов проходило только в системах с общей памятью, особенность их построения позволяет применять эти алгоритмы и в многопроцессорных средах с распределенной памятью.

В качестве критерия оптимальности выступает время получения результатов моделирования. Этот критерий выбран по причине того, что он является одним из основных для данной задачи.

Из данных приведенных таблиц видно, что наименьшее время моделирования параллельный алгоритм показывает при работе в си-

Таблица 1

Зависимость времени моделирования движения ткани из 273 частиц при взаимодействии с объектом из пяти граней (запущено восемь потоков)

Время моделирования, с	Число шагов интегрирования				
	10 000	100 000	1 000 000	10 000 000	100 000 000
При порождении потоков на каждом шаге интегрирования	1,1	9,91	109,81	1084,36	10774,70
При порождении общих потоков для всех шагов интегрирования	1,4	10,25	109,93	1078,24	10710,67

Таблица 2

Зависимость времени выполнения 10 000 итераций моделирования движения ткани из 987 частиц без взаимодействия с объектом от числа запущенных процессов

Время обмена, с	Число процессов				
	1	2	4	6	8
Всей матрицей между процессами	36,64	32,56	40,41	51,49	63,70
Четырьмя строками матрицы между процессами	36,64	19,25	11,33	9,24	8,67

Таблица 3

Зависимость времени выполнения 10 000 итераций моделирования движения ткани из 987 частиц при взаимодействии с объектом из 300 граней от числа запущенных процессов и потоков

Число процессов	1	1	1	1	2	2	4	4	8
Число потоков	1	2	4	8	1	4	1	2	1
Время моделирования, с	71,8	42,5	25,7	16,1	43,4	16,9	28,0	18,2	20,8

стемах с общей памятью, так как отсутствует расход времени на обмен данными между процессами. Следовательно, для достижения максимальной эффективности от применения параллельного алгоритма нужно по-возможности создавать максимальное число потоков, а число процессов сводить к минимуму.

4.2. Реализация параллельных вычислений на графическом процессоре видеоадаптера

Структура графического процессора такова, что позволяет запускать на выполнение практически одновременно большое число потоков (нитей). Как было отмечено ранее, при моделировании поведения ткани движение частиц материала не является абсолютно независимым. В то же время, так как для вычисления скорости частицы на n-м шаге интегрирования требуется учитывать значения координат ее соседних частиц с $(n - 1)$-го шага интегрирования, такая задача подходит для распараллеливания данных на графическом процессоре. Для повышения производительности следует представить данные, передаваемые в память графического процессора, не массивами структур, а классами, содержащими несколько массивов:

```
class Particles_fabric {
public:
float *coord_x;
float *coord_y;
float *coord_z;
...
float *V_x;
float *V_y;
float *V_z;
...
float *F_x;
float *F_y;
float *F_z;
};
```

В программе используются двумерные массивы, которые хранятся в памяти в виде одной связной области. Адрес элемента массива с индексами i, j равен $iN_y + j$, N_y — число столбцов в массиве. Это означает, что элементы, имеющие близкие индексы по оси y, расположены в соседних ячейках памяти [12]. Таким образом, если различные нити обрабатывают различные ячейки по оси y, они будут запрашивать данные из соседних элементов массивов.

Вместе с тем такая форма хранения может приводить к тому, что даже если первый элемент массива выровнен правильно, т. е. его адрес кратен 32 байтам, то нет уверенности в том, что адрес начала следующих строк будет выровнен. Исключение составляет случай, когда соответствующие размеры массива кратны 8, т. е. число байт памяти, занимаемое одной строкой массива, кратно 32 байтам [13]. Поэтому следует отводить память, принудительно расширяя массив так, чтобы выравнивание каждой строки было правильным.

Указатели хранятся в виде обычных указателей float*, так как память для массивов выделяется единообразно, а их параметры, необходимые для выравнивания, зависят только от сетки и могут храниться вместе с ней.

Как и в алгоритме для выполнения на центральном процессоре, между блоками происходит обмен через глобальную память не всеми данными, а только той частью массива, которая содержит информацию о координатах частиц, являющихся соседними для частиц других блоков.

Тестирование представленного алгоритма в зависимости от размера блока проводилось на видеокарте NVIDIA GeForce GTX 760. Следует отметить, что при хронометраже не учитывалось время копирования данных между центральным и графическим процессорами. Результаты тестирования отражены в табл. 4.

Заметим, что время работы ядерной функции не должно превышать 2 с в силу наличия механизма Time-Out Detection and Recovery (TDR), используемого в Microsoft Windows.

На рис. 11 приведены графики для сравнения производительности центрального процессора Intel Core i7-4770K Haswell (запуск задачи в 8 потоков) и гра-

Таблица 4

Зависимость времени выполнения 10 000 итераций моделирования движения ткани из 4056 частиц от размера блока

Размер блока	Не блочный	4×4	8×8	16×16	32×32
Время выполнения, с	16,51	91,1	16,8	11,2	10,3

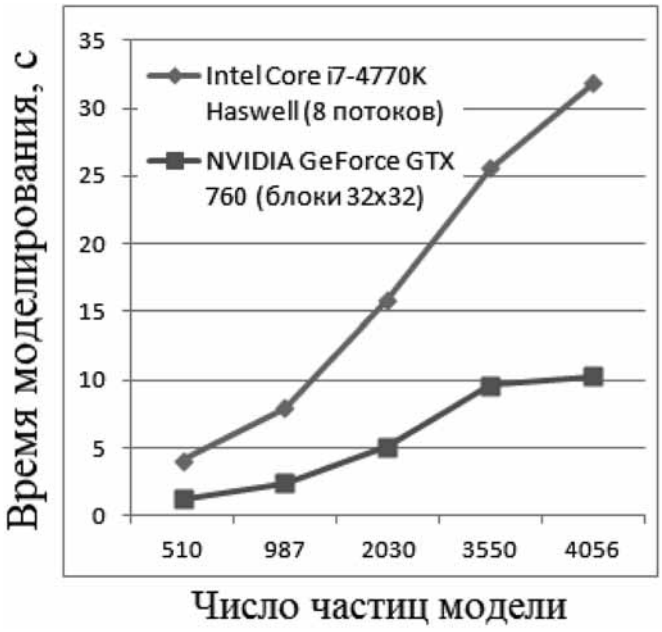


Рис. 11. Сравнение производительности центрального процессора и графического процессора видеокарты в задаче моделирования поведения ткани

фического процессора видеокарты NVIDIA GeForce GTX 760 (размер блока 32×32) в задаче моделирования поведения ткани в зависимости от числа частиц модели при вычислении 10 000 итераций.

Как видно на представленных графиках, использование графического процессора видеокарты NVIDIA GeForce GTX 760 в задаче моделирования поведения ткани позволяет добиться ускорения вычислений в 2,5—4 раза по сравнению с осуществлением расчетов на многоядерном центральном процессоре Intel Core i7-4770K Haswell. Стоит отметить и тот факт, что время вычислений напрямую зависит от характеристик тестируемого оборудования и в каждом конкретном случае может отличаться от данных, приведенных в этой статье. Однако общая тенденция ускорения расчетов при использовании графического процессора видеокарты должна сохраниться.

Заключение

В статье описаны методы параллельной обработки данных с использованием центрального и графического процессоров. Данные алгоритмы были протестированы в различных вычислительных системах для разных видов деталей и форм объектов.

Для тестирования был разработан комплекс программных средств, позволяющий выполнять все этапы компьютерного моделирования сборки изделий из ткани, а также проводить численную и визуальную оценку эффективности сборки через итоговый расчет растяжений и выделение цветом зон напряженности материала. Данный программный комплекс был разработан на языке C++ в среде Microsoft Visual Studio 2012. Для реализации параллельной обработки данных на центральном процессоре для систем с общей и распределенной памятью используются интерфейсы параллельного программирования OpenMP и MPI соответственно. Для обработки данных с использованием ресурсов графического процессора видеоадаптера применяется технология CUDA.

Результаты, приведенные в работе, представляют интерес для развития информационных технологий в области геометрического моделирования и проектирования. Научной новизной в статье являются представленные алгоритмы параллельной обработки данных в процессе моделирования для описанной дискретной модели ткани.

Данная работа будет интересна специалистам в области создания систем автоматизации геометрического моделирования и проектирования, систем виртуальной реальности и компьютерной помощи, специалистам швейной промышленности, а также дизайнерам одежды и мебельного производства.

Список литературы

1. **Terzopoulos D., Platt J., Barr A., Fleischer K.** Elastically deformable models // Computer Graphics. 1987. Vol 21, N 4. P. 205—214.

2. **Weil J.** The synthesis of cloth objects // In Proceedings of SIGGRAPH 86. Computer Graphics Proceedings, Annual Conference Series. 1986. Vol. 20. P. 49—54.

3. **Rudomin I. J.** Simulating Cloth using a Mixed Geometry-Physical Method. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1990.

4. **Kunii T. L., Gotoda H.** Singularity theoretical modeling and animation of garment wrinkle formation process // The Visual Computer. 1990. Vol. 6, N 6. P. 326—336.

5. **Hockney W., Eastwood J.** Computer Simulation Using Particles. McGraw-Hill, New York, 1981.

6. **Бузов Б. А., Алыменкова Н. Д.** Материаловедение в производстве изделий легкой промышленности (швейное производство): учеб. для вузов. М.: Академия, 2004. 448 с.

7. **Breen D. E., House D. H.** Cloth Modeling and Animation / Ed. by D. H. House, D. E. Breen. A K Peters, Ltd. Natick, Massachusetts, 2000. 344 p.

8. **Kawabata S.** The Standardization and Analysis of Hand Evaluation, Textile Machinery Society of Japan. Osaka, Japan, 1980.

9. **Ландовский В. В., Фроловский В. Д.** Исследование методов интегрирования дифференциальных уравнений в задаче моделирования поведения ткани на основе метода частиц // Сибирский журнал вычислительной математики. 2006. Т. 9. С. 287—298.

10. **Хьюз К., Хьюз Т.** Параллельное и распределенное программирование на C++: Пер. с англ. М.: Вильямс, 2004. 672 с.

11. **Корнеев В. Д.** Параллельное программирование кластеров: учеб. пос. Новосибирск: Изд-во НГТУ, 2008. 312 с.

12. **Сандерс Дж., Кэндрот Э.** Технология CUDA в примерах: введение в программирование графических процессоров: Пер. с англ. М.: ДМК Пресс, 2011. 232 с.

13. **Боресков А. В., Харламов А. А.** Основы работы с технологией CUDA. М.: ДМК Пресс, 2011. 232 с.

I. E. Landovskaya, Postgraduate Student**,** e-mail: nairy@rambler.ru,
V. D. Frolovsky, Professor**,** e-mail: frolovskij@corp.nstu.ru,
V. V. Landovsky, Associate Professor, e-mail: landovskij@corp.nstu.ru,
Novosibirsk State Technical University

Fabric Materials Deformation Simulation by Using the Simultaneous Processing Method

The simulation of the fabric materials behavior on the solid plane—bounded object's surface is considered in the paper. The fabric is represented as a discrete model that can virtually to reproduce real material's properties with high accuracy. The item outline (pattern) is used as an initial data to construct item grid model. Commonly, that outline has the form of any polygon. The grid has to be uniform with square-shaped cells so the active forces occurring in the simulating product were brought in modulus and direction to the forces of the real material that occurs with its deformation. Methods for experimental investigation of the fabric deformation properties are described. Expressions for the strain energy that occurs between the fabric discrete model particles are presented. These expressions take into account experimental data. For simulation the physical methods have been used, namely, the particle method. To solve the differential equation system the leapfrog scheme has to be used. To describe the developed parallel processing algorithm for paralleling the calculations on the central processing unit both for shared-memory systems and for distributed memory systems are presented. Implementation method of parallel computing on graphics processor is presented. The results of numerical experiments for described parallel processing algorithms are provided. Also, the fitting on the dress form of the cotton fabric base construction results are shown. The tension areas of the material are highlighted.

Keywords: *fabric materials simulation, method of parallel computing, process, thread, calculations on the central processing unit, computing on graphics processing unit, fabric deformation properties, sampling of the free-form items, particle method, leapfrog scheme*

References

1. **Terzopoulos D., Platt J., Barr A., Fleischer K.** Elastically deformable models, *Computer Graphics*, 1987, vol. 21, no. 4, pp. 205—214.

2. **Weil J.** The synthesis of cloth objects. *Proceedings of SIGGRAPH 86. Computer Graphics Proceedings, Annual Conference Series*, 1986, vol. 20, pp. 49—54.

3. **Rudomin I. J.** Simulating Cloth using a Mixed Geometry-Physical Method. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1990.

4. **Kunii T. L., Gotoda H.** Singularity theoretical modeling and animation of garment wrinkle formation process. *The Visual Computer*, 1990, vol. 6, no. 6, pp. 326—336.

5. **Hockney W., Eastwood J.** *Computer Simulation Using Particles.* McGraw-Hill, New York, 1981.

6. **Buzov B. A., Alymenkova N. D.** *Materialovedenie v proizvodstve izdelij legkoj promyshlennosti (shvejnoe proizvodstvo)*: Uchebnik dlja stud. vyssh. uchebn. zavedenij (Material science for light industry (garment manufactures): Textbook for students of the universities), Moscow, Akademija, 2004. 448 p. (in Russian).

7. **Breen D. E., House D. H.** *Cloth Modeling and Animation* / Ed. by D. H. House, D. E. Breen. A K Peters, Ltd. Natick, Massachusetts, 2000. 344 p.

8. **Kawabata S.** *The Standardization and Analysis of Hand Evaluation.* Textile Machinery Society of Japan. Osaka, Japan, 1980.

9. **Landovsky V. V., Frolovsky V. D.** Issledovanie metodov integrirovanija differencial'nyh uravnenij v zadache modelirovanija povedenija tkani na osnove metoda chastic (Investigation of the techniques of the difference equation integration in the problem of the simulation of the fabric materials behavior by particle method). *Sibirskij Zhurnal Vychislitel'noj Matematiki*, 2006. vol. 9, pp. 287—298 (in Russian).

10. **Hughes C., Hughes T.** *Parallel'noe i raspredelennoe programmirovanie na C++.* Per. s angl. (Parallel and distributed programming using C++). Moscow, Vilyams, 2004. 672 p. (in Russian).

11. **Korneev V. D.** *Parallel'noe programmirovanie klasterov*: uch. posobie (Cluster's parallel programming: study guide). Novosibirsk: Izd-vo NSTU, 2008. 312 p. (in Russian).

12. **Sanders J., Kandrot E.** *Tekhnologiya CUDA v primerakh: vvedenie v programmirovanie graficheskikh protsessorov:* Per. s angl. (CUDA by example. An introduction to general-purpose GPU programming). Moscow, DMK Press, 2011. 232 p. (in Russian).

13. **Boreskov A. V., Kharlamov A. A.** *Osnovy raboty s tekhnologiej CUDA.* (Basic operations with CUDA technology). Moscow, DMK Press, 2011. 232 p. (in Russian).

УДК 519.72

А. М. Кукарцев, ст. препод., e-mail: amkukarcev@yandex.ru,
А. А. Кузнецов, д-р физ.-мат. наук, проф., e-mail: kuznetsov@sibsau.ru
Сибирский государственный аэрокосмический университет
имени академика М. Ф. Решетнева, г. Красноярск

О конструктивном представлении группы Джевонса для инженерно-технических решений обработки информации[1]

*Действия группы Джевонса на множестве бинарных векторов и булевых функций позволяют строить инженерно-технические решения для представления и обработки информации. В статье рассмотрено конструктивное представление группы Джевонса для этого класса прикладных задач. Приведены правила вычисления для разработки программной, аппаратной и программно-аппаратной реализаций и необходимые доказательства их корректности. Предлагаемые правила были успешно апробированы при решении уравнений действия группы Джевонса на множестве булевых функций. Рассматриваемый материал статьи является ядром разработанной библиотеки программных функций "domain operations processor" (или **dop**), в которой реализованы указанные в статье правила вычисления. Библиотека распространяется под лицензией GNU Lesser General Public License.*

Ключевые слова: *представление информации, группа Джевонса, действие группы на множестве, бинарные векторы, булевы функции, вычислительные средства*

Введение

Инженерно-технические решения (далее — ИТР) для создания, обработки, хранения и передачи информации основываются на способах представления информации. Их можно условно разделить на две группы. К первой группе относят способы, основывающиеся на представлении информации как комбинации тех или иных символов того или иного алфавита. Это направление активно развивалось с начала XX века и сегодня известно как теория информации [1]. Отличительной особенностью такого представления информации является независимость символов друг от друга и, как следствие, нечувствительность алгоритмов обработки к перестановке символов даже при потере смысловой нагрузки. Другими словами, не учитывается семантика информации.

С появлением ЭВМ и развитием информационных систем в конце XX века появилось смежное направление. В этой группе информация представляется как комбинация связанных (зависимых) символов того или иного алфавита и отождествляется с некоторой математической функцией. Причем символы являются значениями такой функции. На этом представлении основываются современные методы обработки графической (например, JPEG), звуковой и видеоинформации [2]. В таком представлении символы связанны и есть требования к семантике информации. Эти методы применимы только к некоторым специфичным видам информации, например к графике. Особенностью также являются так называемые потери информации при обработке в силу того, что применяются не дискретные функции.

В настоящей работе рассмотрен способ представления информации, относящийся ко второй группе, и формулируется математическое ядро для разработки ИТР на основе такого представления. Суть представления заключается в следующем. Бинарный вектор (далее — БВ), длина которого кратна n-й степени двойки, сопоставляется с булевой функцией (далее — БФ) n аргументов [3, 4]. При этом БВ является столбцом значений БФ и задано соответствие между аргументами БФ и номерами бит БВ. Если зафиксировать порядок следования аргументов БФ, то сами аргументы рассматриваются как БВ и как двоичные числа. Последнее позволяет связать БФ и БВ.

После такого связывания можно анализировать и преобразовывать БВ, применяя методы теории БФ. Нетрудно видеть, что появляется естественная эквивалентность на множестве БВ. Если переставить и/или инвертировать аргументы БФ, то ее формула

[1] Работа выполнена при поддержке гранта Президента РФ (проект МД-3952.2015.9).

(как упорядоченная комбинация булевых операций) неизменна. Откуда из эквивалентности БФ следует эквивалентность БВ.

Для разработки алгоритмов преобразования информации и последующих ИТР одного лишь представления недостаточно. Нужен математический аппарат, который связывает БВ, БФ, перестановки и инверсии аргументов БФ и элементов БВ. Центральным объектом при этом является алгебраическая группа (далее — группа) Джевонса [3, 5], действующая на множестве БВ и БФ.

Работа с группой Джевонса и ее действиями на множествах БВ и БФ сложна. Это обусловлено многими причинами. Во-первых, размер объектов, на которых действует группа Джевонса, экспоненциально зависит от степени самой группы. Поэтому при построении/отбраковке гипотез расчет объектов без использования ЭВМ даже для небольших значений показателей может занимать десятки минут и часы. Во-вторых, действия проводятся над однообразными объектами, в частности, для булевых функций такими объектами будут ее значения — нули/единицы. В результате объективно допускаются многие ошибки при вычислениях. Поэтому целесообразно использовать ЭВМ, но появляется проблема представления объектов в ЭВМ и их интерпретация человеком. В-третьих, сама группа Джевонса, определяемая как внешнее полупрямое произведение своих подгрупп [3], может быть представлена не единственным способом. Существуют различные методы, но для описания действий над БВ и БФ предпочтительнее использовать полупрямое произведение, потому что такое представление непосредственно отражает целевую задачу. Как известно, внешнее полупрямое произведение может быть задано разными способами [6]. Далее, в основной части статьи, будет показано, что для множеств БВ и БФ группу Джевонса можно представить минимум двумя альтернативными корректными способами. В-четвертых, природа рассматриваемых объектов такова, что при вычислениях аргументы многих операций меняются местами, что обусловлено различными причинами. Например, последовательное действие двух элементов группы Джевонса на БФ эквивалентно действию элемента-произведения на БФ без изменения порядка. Но при таком действии (из свойств действий на самих функциях) должна осуществляться перестановка сомножителей. При построении аналитических соотношений, описывающих модель обработки информации до ее реализации в ИТР, нужно отслеживать такие перестановки. В результате правила вычислений плохо согласуются с человеческой интуицией, что также порождает ошибки.

Цель настоящей статьи — предложить математический аппарат для работы с представлением информации как с БФ, необходимый для разработки соответствующих ИТР. В работе конструктивно определяется (для конкретного гомоморфизма [6]) группа Джевонса, удобная для действий на множествах БВ и БФ. Сами действия в силу ограничения объема статьи рассмотрены только для обоснования выбора гомоморфизма группы Джевонса. Приведены необходимые доказательства, подтверждающие обоснованность сформулированных правил вычисления, и даны рекомендации по их применению. Такие рекомендации будут полезны не только специалистам, начинающим работать с действиями группы Джевонса на множествах БВ и БФ, но и, в силу специфики рассматриваемых объектов, профессионалам.

Предлагаемые правила были успешно применены для решения некоторых уравнений действия группы Джевонса на множестве БФ, рассматриваемых в работах [5, 7, 8]. Без использования данных правил решение таких уравнений тривиальными алгоритмами требует экспоненциального времени работы. Назначение статьи — представление решения технических трудностей реализации действий группы Джевонса на множестве БВ и БФ. Поэтому решения указанных уравнений не приведены.

В заключении дано описание средств обработки рассмотренных операций. В тексте настоящей статьи явно показана сложность работы в этой предметной области. Поэтому были разработаны узкоспециализированная библиотека функций [9] и программная библиотека "domain operations processor" (или **dop**), в которой реализовано ИТР, отражающее рассмотренные в настоящем изложении правила вычисления.

Представляемые материалы статьи, включая приведенные обозначения, являются описанием ядра библиотеки **dop**. Фактически в статье не приведен новый математический теоретический материал. Статья раскрывает внутренние механизмы библиотеки **dop** и предназначена для исследователей в области теории информации и инженеров. Приведенные правила построения вычислений в рассматриваемой предметной области являются практическим руководством, позволяющим отслеживать связи между целевыми объектами, как при построении аналитических расчетов, так и при массивной экспериментальной апробации гипотез.

Базовые операции

Формулируемые правила вычисления предназначены для построения ИТР с применением ЭВМ, поэтому используется числовая нотация индексов (меньшие индексы правее) для векторов и отсчет ведется от нуля.

Пусть $E = \{0, 1\}$ — множество, состоящее из двух элементов. Декартовое произведение этого множества на себя определим как $E^n = \underbrace{E \times \cdots \times E}_{n}$. Под бинарным вектором длины n будем понимать элемент множества E^n. Для координат множителей будем использовать классическую числовую нотацию L2R (left to right), т. е. бóльшие координаты находятся

левее. Обозначим произвольный элемент такого множества как $x \in E^n : x = \{x_{n-1},\ldots,x_i,\ldots,x_0\}$.

Используемые в настоящей статье символика и нотация однозначно соответствуют формату входных и/или выходных данных библиотеки **dop**. Здесь и далее по тексту, а также в библиотеке **dop** фигурные скобки применяют для обозначения БВ, а круглые — для подстановок. Это необходимо для семантического разделения сущностей при эксплуатации **dop**.

Определим операцию на множестве E^n как сложение соответственных координат по модулю 2. Пусть $x, y, z \in E^n$ и $z = x \oplus y$, тогда $z = \{x_{n-1} \oplus y_{n-1},\ldots,x_i \oplus y_i,\ldots,x_0 \oplus y_0\}$. Обозначим множество E^n с операцией $\oplus$ как E_n, это необходимо для семантического разделения сущностей в библиотеке **dop**. Полученная алгебраическая структура является группой (более подробно о группах см. в работах [3, 6]).

Подстановкой [10] степени n будем называть объект вида $\pi = \begin{pmatrix} n-1 & \ldots & 0 \\ i_{n-1} & \ldots & i_0 \end{pmatrix}$. Причем в верхней и нижней строках находятся различные числа от 0 до $n-1$. Подстановка задает биективное отображение множества целых неотрицательных чисел от 0 до $n-1$ на себя.

Нотация подстановок выбрана таким образом, чтобы соответствовать индексам бинарных векторов. При подготовке исходных данных и интерпретации результата такой способ не очень удобен для человека. Но проведенные несколько тысяч практических вычислений показали эффективность такого представления при решении конкретных задач, связанных с действием группы Джевонса на множестве БВ и БФ. Под эффективностью понимаются трудозатраты в связке человек — машина, т. е. от формулировки задачи до получения конечных результатов решений. Также был рассмотрен и традиционный способ задания (записи) подстановок, но практика показала его неэффективность в силу появления путаницы между индексами векторов и значениями подстановки при интерпретации исходных данных и результатов человеком.

Под произведением двух подстановок будем понимать последовательное применение этих подстановок к множеству. Другими словами, результат первой подстановки является исходным значением для второй. Все множество подстановок степени n с операцией умножения обозначим S_n. Более подробно групповые свойства S_n описаны в работах [6, 10].

Действие группы подстановок на множестве бинарных векторов

Группа подстановок S_n может действовать на множестве БВ путем перестановки ее координат. Но такое действие можно задать по-разному. Как будет показано далее, в зависимости от определения правила действия элемента симметрической группы на БВ будут формироваться различные результаты и правила вычисления. Помимо действия на БВ, можно построить действие элементов симметрической группы на БФ [3, 5]. Далее рассмотрим два набора правил вычисления действия группы S_n на БВ.

Обозначим два набора правил вычисления как тип А и тип Б. Далее рассмотрим свойства обоих типов по отношению друг к другу.

Действием типа А элемента группы $\pi \in S_n$ на БВ $x \in E^n$ будем называть вычисление результата $x' \in E^n$: $x' = x^\pi$ по следующему правилу: в верхней строке подстановки находятся "старые" индексы, а в нижней — "новые".

$$x'_{\pi(i)} = x_i. \qquad (1.А)$$

Действием типа Б элемента группы $\pi \in S_n$ на БВ $x \in E^n$ будем называть вычисление результата $x' \in E^n : x' = x^\pi$ по следующему правилу: в верхней строке подстановки находятся "новые" индексы, а в нижней — "старые".

$$x'_i = x_{\pi(i)}. \qquad (1.Б)$$

Типы вычислений А и Б похожи. Преимущество типа А заключается в том, что операнды не меняют свой порядок при вычислении действия-результата на БВ. Но при действии на БФ это нарушается. В свою очередь, тип Б меняет местами множители при вычислении действия-результата на БВ (что нежелательно), но при действии на БФ перестановки операндов не происходит. По этой причине в зависимости от того, что является целевым объектом — БВ или БФ, невозможно выбрать ни тип А, ни тип Б. Более того, действия подстановки на БВ разных типов связаны как действия обратных элементов. Но при переходе к производным объектам, например к БФ, связывающее соотношение становится более сложным. Поэтому целесообразно рассматривать независимо оба типа.

В результате пользователю библиотеки **dop** в зависимости от задачи предоставляется выбор типа правил вычислений. Нужно также указать тип для входных и/или выходных данных при обработке с помощью библиотеки **dop**.

Большинство последующих выводов включают действие по формулам (1.А) и (1.Б) абстрактно. Под "абстрактно" понимается свойство виртуальной полиморфности при наследовании классов в объектно-ориентированном программировании [11]. То есть независимо от типа преобразований приведенные формулы допускают полиморфное обобщение (виртуализация [11]), но конкретные расчетные формулы должны определяться при задании типа преобразования.

Примем за основу следующее суждение: формулируемые законы вывода сохраняют классические математические объекты, в частности произведения (суммы) элементов внутри групп, и **только при действиях** правила вычисления могут становиться неестественными. Тогда сформулируем правила композиций действия нескольких подстановок на БВ по типу А и по типу Б.

Лемма 1.А (О действии типа А). Если подстановки $\pi, \rho \in S_n$ действуют последовательно (π и затем ρ) на элемент $x \in E^n$, то результат действия эквивалентен действию подстановки-произведения $\pi\rho$ или в символьном виде $\left(x^{\pi}\right)^{\rho} = x^{(\pi\rho)}$.

Доказательство. Определим $x' = x^{\pi}$ и $x'' = (x')^{\rho}$, и пусть $i' = \pi(i)$. Тогда по формуле (1.А) получим $x'_{\pi(i)} = x_i = x'_{i'}$ и подставим в $x''_{\rho(i')} = x'_{i'}$. Последнее соотношение также отражает (1.А) независимо от того, какой символ выбран для индекса. Тогда вычислим $x''_{\rho(\pi(i))} = x'_{\pi(i)} = x_i = x''_{(\pi\rho)(i)}$, или при переходе от индексной к векторной форме записи получим $\left(x^{\pi}\right)^{\rho} = x^{(\pi\rho)}$. Что и требовалось доказать.

Лемма 1.Б (О действии типа Б). Если подстановки $\pi, \rho \in S_n$ действуют последовательно (π и затем ρ) на элемент $x \in E^n$, то результат действия эквивалентен действию подстановки-произведения $\rho\pi$ или в символьном виде $\left(x^{\pi}\right)^{\rho} = x^{(\rho\pi)}$.

Доказательство. Определим $x' = x^{\pi}$ и $x'' = (x')^{\rho}$, и пусть $i' = \rho(i)$. Тогда по формуле (1.Б) получим $x''_i = x'_{\rho(i)} = x'_{i'}$ и подставим в $x'_{i'} = x_{\pi(i')}$. Последнее соотношение также отражает (1.Б) независимо от того, какой символ выбран для индекса. Тогда вычислим $x''_i = x'_{i'} = x_{\pi(i')} = x_{\pi(\rho(i))} = x_{(\rho\pi)(i)}$, или при переходе от индексной к векторной форме записи получим $\left(x^{\pi}\right)^{\rho} = x^{(\rho\pi)}$. Что и требовалось доказать.

Действие типа А сохраняет естественный порядок операций при действии группы подстановок на БВ.

Так как в одном выражении указываются как групповые операции, так и действия, например $x^{\pi\rho}$, то возникает неоднозначность в приоритете выполнения операций. Для определенности примем, что действие имеет минимальный приоритет, т. е. для $x^{\pi\rho}$ нужно сначала вычислить произведение $\pi\rho$, а только затем действие над x результатом произведения. Если необходим другой порядок выполнения операций, то условимся использовать скобки для изменения приоритета. Например, $(x^{\pi})^{\rho}$ обозначает последовательное применение действий, сначала π над последовательностью, а затем ρ над результатом первого действия.

Определение группы Джевонса

Группа E_n также может действовать на множестве БВ. Сам БВ длины n является элементом группы E_n. Тогда определим результат действия этой группы на БВ как результат групповой операции исходного вектора и некоторого действующего элемента E_n, а именно положим: $x' = x^z$, где $x, x' \in E^n$ — бинарные вектора и $z \in E_n$ — действующий элемент, $x' = x \oplus z$. Действие, записанное в виде показателя, используется прежде всего для семантики объектов, т. е. чтобы разделить исходные значения и их преобразования. Такая форма записи также позволяет использовать единообразное описание действий над БВ. Далее по тексту примем, что если используется латинский алфавит, то речь идет об элементах E_n, а если греческий — об элементах группы подстановок.

Рассмотрим более детально группы E_n и S_n и то, как элементы этих групп дают способы преобразования БВ.

В специальной литературе [3, 5] можно найти упоминания (либо без указания конкретного гомоморфизма, либо с указанием только одного) о том, что эти группы могут быть объединены в более крупную группу — группу Джевонса D_n. Классически группа Джевонса определяется как полупрямое произведение групп $E_n \leftthreetimes S_n$ [3]. Определение группы Джевонса сводится к тому, что элемент группы Джевонса есть совместное действие всех возможных отрицаний БВ (действие элемента E_n, по сути, не что иное, как выполнение операции отрицания над частями вектора) и всех возможных перестановок элементов вектора. В работе [3] дано именно такое определение.

Основная задача настоящего раздела — описание базовых операций для последующего определения и доказательства более сложных соотношений. В работе [3] дано определение группы Джевонса как группы, действующей на множестве БФ. Как будет показано далее, существуют трудности использования такого задания (представления).

Одной из таких трудностей является перестановочность элементов в произведении (аналогично лемме 1.Б). Трудности, возникающие при вычислении произведений в лемме 1.Б, могут быть решены переопределением самого действия. Но для БФ трудность перестановочности сомножителей является следствием самой природы БФ, а не описанием. По этой причине определение из [3] не может использоваться для настоящей работы. В рамках данной статьи действия группы Джевонса на множестве БФ детально рассмотреть невозможно. Авторы предлагают детальное рассмотрение в последующих работах.

Второй трудностью является сам способ задания группы Джевонса. Она заключается в том, что внешнее полупрямое произведение групп может быть задано многими способами и в результате давать различные (неизоморфные между собой, например прямое произведение) группы [6].

Рассмотрим классическое определение внешнего полупрямого произведения некоторых групп N и H. Определим операцию для двух элементов группы-произведения (n_0, h_0) и (n_1, h_1), где $n_0, n_1 \in N$, $h_0, h_1 \in H$ [6]:

$$(n_0, h_0) \circ (n_1, h_1) = \left(n_0 \psi_{h_0}(n_1), h_0 h_1\right). \quad (2)$$

Здесь $\psi_{h_0}(n_1)$ — автоморфный образ элемента n_1, причем гомоморфизм зависит от элемента h_0. И в зависимости от выбора гомоморфизма ψ могут быть получены различные группы-произведения, в том числе вырожденный случай — прямое произведение (ψ — тождественно). Полученные различные полупрямые произведения могут называться группой Джевонса, так как для внешнего полупрямого

произведения выбор конкретного ψ — это вопрос задания. Отсюда заключаем, что нужны основания для выбора конкретного ψ для реализации действий группы Джевонса на множествах БВ и БФ.

Одним из таких оснований является связь двух групп по действию на вектор. Далее покажем на примере, что исключительно действие на вектор не позволяет выбрать гомоморфизм. Рассмотрим уравнение $x^{\pi} = x'$, где $x, x' \in E^n$, $\pi \in S_n$. Решить это уравнение относительно π — значит найти все решения или показать, что их нет. Рассмотрим случай, при котором нет решений. Так как x, x' — векторы одинаковой длины, а π меняет местами элементы вектора, то необходимо и достаточно, чтобы они содержали равное число нулей и единиц. Поэтому не имеет смысла рассматривать случаи отсутствия решений. Если же уравнение разрешимо относительно π, то можно построить эквивалентное уравнение $x^z = x'$, где $z \in E_n$, так как элементы x, x' множества E^n являются и элементами группы E_n, а потому такие уравнения всегда разрешимы. Другими словами, всякое действие группы S_n сводимо к действию E_n. Поэтому относительно действия групп E_n и S_n на множестве E^n ни одно определение ψ не конструктивно для целевой задачи. Основания выбора ψ для настоящей работы — действия над БФ.

Сначала рассмотрим искомый гомоморфизм абстрактно [11], а после дадим конкретные вычислительные формулы и доказательства. Группа Джевонса индуцирует действие на множестве БФ так, что все множество БФ разбивается на непересекающиеся классы. Для этого определим некоторую группу β_n, которая индуцирует на множестве n-местных БФ действие, эквивалентное действию группы Джевонса, т. е. разбивает все множество БФ на такие же непересекающиеся классы. Пусть также группа β_n содержит элементы одной природы и раскладывается во внутреннее полупрямое произведение своих подгрупп: $\beta_n = B_n \leftthreetimes T_n$.

Так как произведение указанных групп внутреннее, то гомоморфизм из соотношения (2) будет внутренним и будет задаваться сопряжениями элементов естественно. Определим биекции (необязательно изоморфизмы) $E_n \to B_n$, $S_n \to T_n$ и $D_n \to \beta_n$.

Термины "биекция" и "природа" применяют вместо изоморфизма и эквивалентности потому, что, как будет показано далее, наряду с изоморфными вложениями групп появляются и антиизоморфные. Антиизоморфные вложения не нарушают требование разбиения множества БФ на непересекающиеся классы, эквивалентные разбиению с помощью группы Джевонса, поэтому допустимы.

В результате таких построений получится группа, которая задается независимо от БФ и явно с ними не связана, но в то же время индуцирует такое же действие, как и группа Джевонса. А их действия на БВ и БФ будут задаваться независимо от них самих. Причем правила вычисления действий вполне могут нарушать классические групповые соотношения, так же как в леммах 1.А и 1.Б. Перейдем к конкретному описанию группы β_n и к обоснованию вышеописанных требований. Сначала дадим некоторые опорные суждения.

Сопоставим с каждым элементом множества E^n некоторое неотрицательное целое число, число и вектор будем обозначать одинаковыми символами, так как вектор, по сути, есть двоичная форма записи искомого числа:

$$\forall x \in E^n, \exists! x \in Z_+ : x = \sum_{i=n-1}^{0} x_i 2^i. \qquad (3)$$

Отображение (3) является биекцией и верно $0 \leqslant x < 2^n$. Порядок индексов определяется в единой нотации L2R. Тогда определим действие элементов группы E_n и S_n над элементами множества Z_+ по аналогии с соответствующими им БВ. Теперь покажем, что действие симметрической группы на группе БВ есть автоморфизм на группе E_n.

Теорема 1 (Об автоморфизме). Отображение $E_n^{\pi} \to E_n$, $\forall \pi \in S_n$ есть автоморфизм независимо от типа преобразования А или Б.

Доказательство. Сначала докажем гомоморфность, а затем биективность. Пусть $x, y, z \in E_n$: $z = x \oplus y$, тогда если отображение — автоморфизм, то верно $z^{\pi} = x^{\pi} \oplus y^{\pi}$.

Применим действие к компоненту i левой части $(z_i)^{\pi}$. Оно раскроется по типу А как $z_{\pi(i)}$ и по типу Б как $z_{\pi^{-1}(i)}$. Из определения внутригрупповой операции в E_n имеем $z_i = x_i \oplus y_i$, тогда получим для типа А соотношение $x_{\pi(i)} \oplus y_{\pi(i)}$, а для типа Б — $x_{\pi^{-1}(i)} \oplus y_{\pi^{-1}(i)}$. Перейдем к унифицированной форме действия (в виде леммы 1.А или 1.Б) как $(z_i)^{\pi} = (x_i)^{\pi} \oplus (y_i)^{\pi}$. Для типа А это верно по определению, а для типа Б верно из $\forall \pi, \exists! \pi^{-1} : \pi\pi^{-1} = \pi^{-1}\pi = e$, откуда заключаем, что отображение сохраняет операцию независимо от типа А или Б.

Из общей алгебры верно [6], что любая подстановка порождается транспозициями и даже транспозициями вида $(0, i)$. Рассмотрим, как действует на все множество E^n такая транспозиция. Разобьем все множество группы E_n на классы по следующему правилу. Значения координат БВ внутри каждого класса совпадают, кроме координат 0 и i. В каждом таком классе будет по четыре вектора, а всего классов 2^{n-2}. Транспозиция переставляет координаты 0 и i, поэтому ее действие будет локализовано внутри классов. Значения координат, которыми отличаются векторы внутри класса, являются: 00, 01, 10, 11. В случае 00 и 11 отображение будет тривиально. В случае 01 и 10 векторы будут отображаться друг в друга, откуда заключаем, что транспозиция отображает E_n биективно на себя. Тогда для типа А по лемме 1.А заключаем, что $E_n^{\pi} \to E_n$ биекция. Для типа Б по лемме 1.Б транспозиции, образующие π, будут действовать в обратном порядке, но будут биекциями, т. е. по типу Б отображение $E_n^{\pi} \to E_n$ тоже биекция. Следовательно, отображение $E_n^{\pi} \to E_n \forall \pi \in S_n$ —

биекция, сохраняющая операцию независимо от типа преобразования А или Б. Что и требовалось доказать.

Теорема 1 показывает, что действие любого элемента симметрической группы по формуле (1.А) или (1.Б) на всю группу E_n есть автоморфизм. Но отображение $E_n^{\pi} \to E_n \forall \pi \in S_n$ не является автоморфизмом для искомого гомоморфизма ψ в (3). Но он с ним связан. Покажем это.

Действие элемента группы $z \in E_n$ на $x \in E^n$ в виде (3) можно записать следующим образом:

$$x^z = \sum_{i=n-1}^{0} (x_i \oplus z_i) 2^i. \qquad (4)$$

При этом нет необходимости перехода к вектору.

В свою очередь, для действия элемента группы $\pi \in S_n$ в зависимости от типа преобразования по (1.А) или по (1.Б) в виде (3) получим:

$$x^{\pi} = \sum_{i=n-1}^{0} x_{\pi^{-1}(i)} 2^i = \sum_{i=n-1}^{0} x_i 2^{\pi(i)}; \qquad (5.А)$$

$$x^{\pi} = \sum_{i=n-1}^{0} x_{\pi(i)} 2^i = \sum_{i=n-1}^{0} x_i 2^{\pi^{-1}(i)}. \qquad (5.Б)$$

Преобразование типа А неудобно для "быстрого счета" и для записи в виде действия на вектор. Правая часть (5.А) позволяет устранить эту трудность.

Для удобства описания следующих соотношений определим $k = 2^n$. Пусть $0 \leqslant j < k$, тогда определим отображения:

$$\varphi_z(j) = j^z, \forall z \in E_n; \qquad (6)$$

$$\varphi_{\pi}(j) = j^{\pi}, \forall \pi \in S_n. \qquad (7)$$

Так как j пробегает все значения от 0 до $k-1$ и j^z — биекция как групповая операция в E_n, а j^{π} — биекция согласно теореме 1, то отображения φ_z и φ_{π} сами являются биекциями множества чисел от 0 до $k-1$ на себя. Другими словами, φ_z и φ_{π} — подстановки симметрической группы S_k. Обозначим множество элементов φ_z как B_n и множество элементов φ_{π} как T_n. Покажем, что эти отображения — биекции и сохраняют операцию (включая антиизоморфизмы).

Теорема 2 (Об изоморфизме B_n). Отображение вида (6) $E_n \to B_n$, $z \in E_n$, $\varphi_z \in B_n : \varphi_z(j) = j^z$ есть изоморфизм.

Доказательство. Сохранение операции показывается непосредственно из произведения подстановок. Пусть $z_0, z_1 \in E_n$, тогда вычислим $\varphi_{z_0}\varphi_{z_1}$. Из правил перемножения подстановок имеем $(\varphi_{z_0}\varphi_{z_1})(j) = \varphi_{z_1}(\varphi_{z_0}(j)) = \varphi_{z_1}(j^{z_0}) = (j^{z_0})^{z_1} = j^{z_0 z_1} = \varphi_{z_0 z_1}(j)$ или $\varphi_{z_0}\varphi_{z_1} = \varphi_{z_0 z_1}$. Отсюда заключаем, что операция сохраняется. Из формулы (6) следует, что для любого $z \in E_n$ найдется образ. Тогда покажем, что у такого образа будет только один прообраз. Будем вести доказательство от обратного. Пусть $z_0 \neq z_1$ и $\varphi_{z_0} = \varphi_{z_1}$ или $\varphi_{z_0}(j) = \varphi_{z_1}(j), \forall j \in E^n$, откуда перейдем к тождеству $j^{z_0} = j^{z_1} \Rightarrow j \oplus z_0 = j \oplus z_1$, которое будем преобразовывать внутри группы E_n. Тогда верно $j \oplus j \oplus z_0 = j \oplus j \oplus z_1 \Rightarrow z_0 = z_1$, откуда получаем противоречие. Следовательно, отображение (6) — биекция и сохраняет операцию. Что и требовалось доказать.

Из теоремы 2 заключаем, что $B_n < S_k$ — изоморфное вложение группы E_n в S_k. Полное отображение вектора в подстановку запишем в следующем виде:

$$\varphi_z = \begin{pmatrix} 2^n - 1 & \dots & j & \dots & 0 \\ \sum_{i=n-1}^{0} \overline{z_i} 2^i & \dots & \sum_{i=n-1}^{0} (j_i \oplus z_i) 2^i & \dots & \sum_{i=n-1}^{0} z_i 2^i \end{pmatrix}. \qquad (8)$$

Первый и последний элементы могут быть заданы явно, остальные рассчитываются через j.

Теорема 3 (Об изоморфизме T_n). Отображение вида (7) $S_n \to T_n, \pi \in S_n, \varphi_{\pi} \in T_n : \varphi_{\pi(j)} = j^{\pi}$ есть изоморфизм для типа преобразования А и антиизоморфизм для типа преобразования Б.

Доказательство. Сохранение операции показывается непосредственно из произведения подстановок. Пусть $\pi_0, \pi_1 \in S_n$, тогда вычислим $\varphi_{\pi_0}\varphi_{\pi_1}$. Из правил перемножения подстановок имеем $(\varphi_{\pi_0}\varphi_{\pi_1})(j) = \varphi_{\pi_1}(\varphi_{\pi_0}(j)) = \varphi_{\pi_1}(j^{\pi_0}) = (j^{\pi_0})^{\pi_1}$. Далее по типу А согласно (1.А) имеем $(j^{\pi_0})^{\pi_1} = j^{\pi_0\pi_1} = \varphi_{\pi_0\pi_1}(j)$ или $\varphi_{\pi_0}\varphi_{\pi_1} = \varphi_{\pi_0\pi_1}$, и для типа Б согласно (1.Б) имеем $(j^{\pi_0})^{\pi_1} = j^{\pi_1\pi_0} = \varphi_{\pi_1\pi_0}(j)$ или $\varphi_{\pi_0}\varphi_{\pi_1} = \varphi_{\pi_1\pi_0}$, откуда заключаем, что операция сохраняется, но для типа Б приводит к перестановке образов. Из формулы (7) следует, что для любого $\pi \in S_n$ найдется образ. Тогда покажем, что у такого образа будет только один прообраз. Будем вести доказательство от обратного. Пусть $\pi_0 \neq \pi_1$ и $\varphi_{\pi_0} = \varphi_{\pi_1}$ или $\varphi_{\pi_0}(j) = \varphi_{\pi_1}(j), \forall j \in E^n$. Отсюда перейдем к тождеству $j^{\pi_0} = j^{\pi_1}$, которое будем преобразовывать внутри группы T_n. Тогда верно $(j^{\pi_0})^{\pi_1^{-1}} = (j^{\pi_1})^{\pi_1^{-1}} \Rightarrow (j^{\pi_0})^{\pi_1^{-1}} = j$. Переходя к компонентам j по формуле (1.А) или (1.Б), получим, что $(\pi_0\pi_1^{-1})(i) = i$ или $(\pi_1^{-1}\pi_0)(i) = i$. То есть независимо от типа преобразования получаем тривиальную подстановку и, как следствие, $\pi_0 = \pi_1$, откуда заключаем противоречие. Следовательно, отображение (7) — биекция и сохраняет операцию, но по типу Б приводит к перестановке образов. Что и требовалось доказать.

Из теоремы 3 заключаем, что $T_n < S_k$ — изоморфное (для типа Б — антиизоморфное) вложение группы S_n в S_k. Полное отображение подстановки в подстановку запишем в следующем виде для типов А и Б соответственно:

$$\varphi_{\pi} = \begin{pmatrix} 2^n - 1 & \dots & j & \dots & 0 \\ 2^n - 1 & \dots & \sum_{i=n-1}^{0} j_i 2^{\pi(i)} & \dots & 0 \end{pmatrix}; \qquad (9.А)$$

$$\varphi_\pi = \begin{pmatrix} 2^n-1 & \dots & j & \dots & 0 \\ 2^n-1 & \dots & \sum_{i=n-1}^{0} j_{\pi(i)}2^i & \dots & 0 \end{pmatrix}. \qquad (9.Б)$$

Первый и последний элементы могут быть заданы явно, остальные рассчитываются через j.

Резюме. Определены две группы B_n и T_n, которые являются подгруппами в S_k.

Сконструируем множество β_n как произведения (по групповой операции S_k) всех возможных элементов указанных групп в любых порядках. Для определенности примем, что сначала указывается элемент $b \in B_n$, а затем $t \in T_n$ (далее будет показано, что произведения в обратном порядке сводимы к указанному). Элемент множества β_n запишем как (bt).

Теорема 4 (Об изоморфизме β_n). Множество $\beta_n = B_nT_n$, образованное как теоретико-множественное произведение групп B_n и T_n, есть группа, которая является внутренним полупрямым групповым произведением $B_n \leftthreetimes T_n$.

Доказательство. Сначала покажем, что $B_n \cap T_n = e$. Рассмотрим соотношения (8) и (9.А) или (9.Б). Сравним в общем виде получаемые подстановки по первому и последнему элементам. Для (9.А) или (9.Б) эти элементы неподвижны. В свою очередь, для (8) подвижными будут не только первый и последний элементы, но и все остальные. И только для нейтральных элементов будут получены одинаковые подстановки. Далее покажем, что любое произведение (t_0b_0) сводимо к (b_1t_1). Запишем произведение (t_0b_0) в виде произведения изоморфизмов прообразов (6) и (7). Пусть $\pi \in S_n$ соответствует t_0, а $z \in E_n$ — b_0, и $(t_0b_0) = \varphi_\pi\varphi_z$. Тогда вычислим $(t_0b_0)(j) = \varphi_z(\varphi_\pi(j)) = \varphi_z\left(j^\pi\right) = \left(j^\pi\right)^z$ абстрактно без привязки к типу А или типу Б. Пользуясь тем, что $\left(j^\pi\right)^z = j^\pi \oplus z = j^\pi \oplus \left(z^{\pi^{-1}}\right)^\pi$, и теоремой 1, получим $\left(j^\pi\right)^z = \left(j \oplus z^{\pi^{-1}}\right)^\pi = \left(j^{z^{\pi^{-1}}}\right)^\pi = \varphi_\pi\left(j^{z^{\pi^{-1}}}\right) = \varphi_\pi\left(\varphi_{z^{\pi^{-1}}}(j)\right)$, т. е. $(t_0b_0) = \varphi_{z^{\pi^{-1}}}\varphi_\pi = (b_1t_1)$. Данное выражение показывает, во-первых, что любой элемент представим в виде произведения элементов исходных групп. Во-вторых, из указанного представления и общей ассоциативности симметрической группы следует замкнутость и наличие обратных элементов. То есть β_n — группа. В-третьих, $t_1 = t_0$ и $z^{\pi^{-1}} \in E_n$. Другими словами, любой элемент из β_n можно переставить с элементами B_n. Отсюда заключаем, что $B_n \lhd \beta_n$. В совокупности полученные выводы являются определением полупрямого произведения с нормальным делителем B_n. Что и требовалось доказать.

Как будет показано далее, группа Джевонса может быть представлена более чем одним способом. В свою очередь, группа β_n и ее подгруппы B_n и T_n являются подгруппами группы подстановок S_k и не зависят от типа А или типа Б. Группа β_n **задается независимо от типа**. Она вообще не зависит от групп E_n и S_n и от понятия действия последней на БВ и/или БФ. По этой причине в теореме 4 доказательство дано абстрактно, т. е. использование изоморфизмов (6) и (7) необходимо только для доказательства утверждений теоремы 4, а не для задания самой β_n.

Перейдем к представлению группы Джевонса и вложим ее изоморфно в группу β_n. Обозначим элемент группы Джевонса, отражая внешнее полупрямое произведение $D_n = E_n \leftthreetimes S_n$ как $(z\pi) \in D_n : z \in E_n$, $\pi \in S_n$. Из свойств полупрямых произведений групп следует $(z\pi) = (ze_S)(e_E\pi)$, где e_s и e_E — нейтральные элементы групп S_n и E_n соответственно. Тогда пусть элемент (ze_S) отображается в S_k согласно (6), а элемент $(e_E\pi)$ — согласно (7). Весь же элемент группы Джевонса будет отображаться в их произведение или в символьном виде $(z\pi) = \varphi_z\varphi_\pi$.

Положим $(z_0\pi_0), (z_1\pi_1) \in D_n$, тогда вычислим произведения их образов в β_n или $\varphi_{z_0}\varphi_{\pi_0}\varphi_{z_1}\varphi_{\pi_1}$. Опираясь на соотношения теоремы 4, сначала вычислим произведение внутренних элементов $\varphi_{\pi_0}\varphi_{z_1} = \varphi_{z_1^{\pi_0^{-1}}}\varphi_{\pi_0}$. Далее, согласно теоремам 2 и 3, получим $\varphi_{z_0}\varphi_{\pi_0}\varphi_{z_1}\varphi_{\pi_1} = \varphi_{z_0}\varphi_{z_1^{\pi_0^{-1}}}\varphi_{\pi_0}\varphi_{\pi_1} = \varphi_{z_0z_1^{\pi_0^{-1}}}\varphi_{\pi_0}\varphi_{\pi_1}$. Произведение двух последних элементов раскроется для типов А и Б неодинаково. Для типа А $\varphi_{z_0z_1^{\pi_0^{-1}}}\varphi_{\pi_0}\varphi_{\pi_1} = \varphi_{z_0z_1^{\pi_0^{-1}}}\varphi_{\pi_0\pi_1}$. и для типа Б $\varphi_{z_0z_1^{\pi_0^{-1}}}\varphi_{\pi_0}\varphi_{\pi_1} = \varphi_{z_0z_1^{\pi_0^{-1}}}\varphi_{\pi_1\pi_0}$. Итак, получим, что операция в группе Джевонса задана следующим образом для типов А и Б соответственно.

$$(z_0\pi_0)(z_1\pi_1) = \left(z_0z_1^{\pi_0^{-1}}\pi_0\pi_1\right); \qquad (10.А)$$

$$(z_0\pi_0)(z_1\pi_1) = \left(z_0z_1^{\pi_0^{-1}}\pi_1\pi_0\right). \qquad (10.Б)$$

Отсюда получаем два представления группы Джевонса типа А и типа Б. Обе группы независимо от типа преобразования изоморфны (по построению) группе β_n. Группа Джевонса типа А полностью соответствует определению полупрямого произведения (2). В свою очередь, группа Джевонса типа Б отлична от группы, описываемой формулой (2), так как множитель симметрической группы вкладывается антиизоморфно. Обе операции задаются таким образом, что гомоморфизм ψ из (2) имеет одинаковый виртуальный вид $\psi_{\pi_0}(z_1) = z_1^{\pi_0^{-1}}$. Для каждого конкретного случая необходимо сводить к (1.А) или (1.Б).

Гомоморфизм ψ из соотношения (2) группы Джевонса можно задать в том числе как $\psi_{\pi_0}(z_1) = z_1^{\pi_0}$ и для типа А, и для типа Б. Полученные алгебраические структуры будут являться группами с операциями, отличными от (10.А) и (10.Б), и могут также называться группами Джевонса. Более того, сам такой гомоморфизм корректен в том смысле, что элементы группы S_n действуют естественным образом на элементах группы E_n. Но полученные таким об-

разом группы не будут эквивалентны группе β_n при действии на БФ.

Для расчетов определим формулы для вычисления обратного элемента и сопряженного элемента. Обратный элемент будет абстрактно одинаков для типа А и типа Б. Согласно правилам вычисления [6] обратного элемента полупрямого произведения, имеем:

$$(z\pi)^{-1} = \left(\psi_{\pi^{-1}}\left(z^{-1}\right)\pi^{-1}\right) = \left(\left(z^{-1}\right)^{\left(\pi^{-1}\right)^{-1}}\pi^{-1}\right) = \left(z^{\pi}\pi^{-1}\right). \quad (11)$$

Выражение $z^{-1} = z$ верно, так как порядок элементов группы E_n равен 2. Соотношение (11) абстрактно, т. е. конечные соотношения для типа А и типа Б будут различны (в частности, будут различны части нормального делителя).

Сопряжения элементов в группе Джевонса необходимы для перестановки сомножителей по классическим групповым законам и часто используются в прикладных задачах. Для типов преобразований А и Б будем вести вывод формул сопряженного элемента параллельно. Пусть $(z_0\pi_0)$, $(z_1\pi_1) \in D_n$, тогда вычислим $(z_0\pi_0)^{(z_1\pi_1)}$. Из определения сопряженного элемента получим $(z_1\pi_1)^{-1}(z_0\pi_0)(z_1\pi_1)$, далее, согласно формуле (11), будет верно $\left(z_1^{\pi_1}\pi_1^{-1}\right)(z_0\pi_0)(z_1\pi_1)$. Теперь найдем произведение для типа А и типа Б отдельно. Для типа А, согласно (10.А), получим

$$\left(z_1^{\pi_1}z_0^{\pi_1}\pi_1^{-1}\pi_0\right)(z_1\pi_1) = \left(z_1^{\pi_1}z_0^{\pi_1}z_1^{\left(\pi_1^{-1}\pi_0\right)^{-1}}\pi_1^{-1}\pi_0\pi_1\right) =$$

$$= \left(z_1^{\pi_1}z_0^{\pi_1}z_1^{\pi_0^{-1}\pi_1}\pi_0^{\pi_1}\right) = \left(\left(z_1z_0z_1^{\pi_0^{-1}}\right)^{\pi_1}\pi_0^{\pi_1}\right).$$

Для типа Б, согласно (10.Б) и правилам произведения действий (1.Б), вычислим

$$\left(z_1^{\pi_1}z_0^{\pi_1}\pi_0\pi_1^{-1}\right)(z_1\pi_1) = \left(z_1^{\pi_1}z_0^{\pi_1}z_1^{\left(\pi_0\pi_1^{-1}\right)^{-1}}\pi_1\pi_0\pi_1^{-1}\right) =$$

$$= \left(z_1^{\pi_1}z_0^{\pi_1}z_1^{\pi_1\pi_0^{-1}}\pi_0^{\pi_1^{-1}}\right) = \left(\left(z_1z_0z_1^{\pi_0^{-1}}\right)^{\pi_1}\pi_0^{\pi_1^{-1}}\right).$$

В итоге для типа А и типа Б получим:

$$(z_0\pi_0)^{(z_1\pi_1)} = \left(\left(z_0z_1z_1^{\pi_0^{-1}}\right)^{\pi_1}\pi_0^{\pi_1}\right); \quad (12.А)$$

$$(z_0\pi_0)^{(z_1\pi_1)} = \left(\left(z_0z_1z_1^{\pi_0^{-1}}\right)^{\pi_1}\pi_0^{\pi_1^{-1}}\right). \quad (12.Б)$$

Элементы нормального делителя $\left(z_0z_1z_1^{\pi_0^{-1}}\right)^{\pi_1}$ в (12.А) и (12.Б) совпадают только абстрактно. Выражение $\left(z_0z_1z_1^{\pi_0^{-1}}\right)^{\pi_1}$ нужно раскрывать как (1.А) или (1.Б) для типов А и Б соответственно.

Заключение

Большинство описанных выше операций сложны для ручной обработки. Это обусловлено, во-первых, двойственностью представления группы Джевонса, действующей на множестве БВ и БФ. Во-вторых, наличием нетривиальных подгрупп инерции БВ и БФ в группе Джевонса, т. е. из (1.А) или (1.Б) возможно получение одинакового результата при неодинаковых элементах группы Джевонса. В-третьих, вычисление даже самых простых (при $n > 3$) действий проводится с однообразными значениями (нули и единицы). Это требует повышенного внимания от человека и, как правило, приводит к ошибкам, которые обнаруживают себя только по окончании многоэтапных сложных вычислений.

Для обеспечения возможности работы в вышеописанной предметной области была разработана библиотека "domain operations processor" (или **dop**). Библиотека содержит набор алгоритмов, специализированных для обработки подстановок и их действий на БВ. В отличие от таких систем, как GAP, она не является универсальной для широкого класса задач. Поэтому достигается высокая производительность при вычислениях полным перебором при формировании/отбраковке гипотез в предметной области. Например, для обработки миллиарда элементов из E^n требуется 5...7 мин на одноядерном процессоре 2...3 ГГц. Библиотека распространяется под лицензией GNU LGPL (GNU Lesser General Public License). В ней реализованы все описанные выше операции для обоих типов преобразований (А и Б). Библиотеки **dop** написана на языке C++ в виде системы функций и макросов для архитектуры процессора IA-32. Библиотека является кроссплатформенной, так как не использует программный интерфейс Windows- или POSIX- подобных систем.

Проведя разработку макета библиотеки **dop**, авторы определили, что невозможно построить универсальную библиотеку для всех типов вычислительных комплексов. Для обеспечения гибкости такие механизмы, как процедуры работы с памятью и параллельные вычисления, были вынесены за библиотеку. Такие механизмы должны добавляться для конкретной целевой задачи. Для описанного выше материала библиотека включает в себя: инструменты ввода-вывода объектов, обработки (групповые операции, изоморфизмы, антиизоморфизмы, действия и пр.), перечисления объектов (для построения стендов при исследовании предметной области), средства рандомизации. Другими словами, библиотека является совокупностью функциональных примитивов, каждый из которых реализует описанную в основной части статьи сложную алгоритмическую операцию. Состав библиотеки, включая документацию разработчика (Software Development Kit или SDK), можно получить у авторов настоящей статьи.

Список литературы

1. **Шеннон К.** Работы по теории информации и кибернетике: Пер. с англ. М.: Изд-во иностр. лит., 1963. 830 с.
2. **Сэломон Д.** Сжатие данных, изображений и звука. М.: Техносфера, 2004. 368 с.
3. **Логачёв О. А., Сальников А. А., Ященко В. В.** Булевы функции в теории кодирования и криптологии. М.: МЦМНО, 2004. 470 с.
4. **Марченко С. С.** Замкнутые классы булевых функций. М.: ФИЗМАТЛИТ, 2000. 128 с.
5. **Глухов М. М., Ремизов А. Б., Шапошников В. А.** Обзор по теории k-значных функций. Часть 1. Спр. пособие. М.: Типография в/ч 33965, 1988.
6. **Каргаполов М. И., Мерзляков Ю. И.** Основы теории групп. 2-е изд. М.: Наука, 1977.
7. **Яубайтис Э. А.** Субклассы и классы булевых функций // Автоматика и вычислительная техника. 1974. № 1. С. 1—8.
8. **Golomb S. W.** On classification of Boolean functions // IRE, Trans.circuit theory. 1959. N 6, Spec. Suppl. P. 176—186.
9. **Бентли Дж.** Жемчужины программирования. 2-е изд. СПб.: Питер, 2002. 272 с.
10. **Супруненко Д. А.** Группы подстановок. Минск.: Навука і тэхніка, 1996. 366 с.
11. **Страуструп Б.** Язык программирования C++. Специальное издание. Пер. с англ. М.: Бином, 2011. 1136 с.

A. M. Kukartsev, Senior Lecturer, e-mail: amkukarcev@yandex.ru,
A. A. Kuznetsov, Professor, e-mail: kuznetsov@sibsau.ru,
Siberian State Aerospace University named after academician M. F. Reshetnev, Krasnoyarsk

Constructive Representation of the Jevons Group for Engineering Solutions of Information Processing

Actions of the Jevons group on the set of binary vectors of the Boolean functions allow to build engineering solutions for the submission and processing of information. In this case information will be presented as a binary vector of length 2^n associated with the Boolean function. The binary vector is a column of values of the Boolean function. There is a correspondence between arguments of the function and coordinates of the vector. The correspondence allows analyzing and converting binary vectors using methods of theory of the Boolean functions. So there is a need for mathematical methods and algorithms which connect the Boolean vectors and the Boolean functions. The main object in the solution of this problem is the Jevons group. It is discussed the constructive representation of the Jevons group in the article. We give correct rules of computation which are needed for creating of software and hardware. The proposed rules have been successfully tested in the solution of equations of the Jevons group's action on the set of the Boolean functions. We have developed the library of software functions "domain operations processor" (or dop) based on the rules. The library is licensed under the GNU Lesser General Public License.

Keywords: *information representation, the Jevons group, action on the set, binary vectors, Boolean functions, computational tools*

References

1. **Shennon K.** *Raboty po teorii informatsii i kibernetike.* Per. s angl., Moscow, Izd-vo inostr. lit., 1963. 830 p. (in Russian).
2. **Selomon D.** *Szhatie dannykh, izobrazhenii i zvuka,* Moscow, Tekhnosfera, 2004. 368 p. (in Russian).
3. **Logachev O. A., Sal'nikov A. A., Yashchenko V. V.** *Bulevy funktsii v teorii kodirovaniya i kriptologii,* Moscow, MTsMNO, 2004, 470 p. (in Russian).
4. **Marchenko S. S.** *Zamknutye klassy bulevykh funktsii,* Moscow, Fizmatlit, 2000, 128 p. (in Russian).
5. **Glukhov M. M., Remizov A. B., Shaposhnikov V. A.** *Obzor po teorii k-znachnykh funktsii,* Part 1. Spr. posobie, Moscow, Tipografiya v/ch 33965, 1988 (in Russian).
6. **Kargapolov M. I., Merzlyakov Yu. I.** *Osnovy teorii grupp.* 2nd edition, Moscow, Nauka, 1977 (in Russian).
7. **Yaubaitis E. A.** Subklassy i klassy bulevykh funktsii. *Avtomatika i Vychislitel'naya Tekhnika,* 1974, no. 1, pp. 1—8 (in Russian).
8. **Golomb S. W.** On classification of Boolean functions. *IRE, Trans.circuit theory,* 1959, no. 6, Spec. Suppl, pp. 176—186.
9. **Bentli D.** *Zhemchuzhiny programmirovaniya.* 2-e izd., Saint Petersburg, Piter, 2002, 272 p. (in Russian).
10. **Suprunenko D. A.** *Gruppy podstanovok,* Minsk, Navuka i tekhnika, 1996. 366 p. (in Russian).
11. **Straustrup B.** *Yazyk programmirovaniya C++.* Spetsial'noe izdanie. Per. s angl., Moscow, Binom, 2011, 1136 p. (in Russian).

УДК 004.72:519.2

Д. А. Перепелкин, канд. техн. наук, доц., e-mail: dmitryperepelkin@mail.ru,
М. А. Иванчикова, студент, Рязанский государственный радиотехнический университет

Разработка программного обеспечения моделирования процессов адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи с различными зонами покрытия абонентов

Предложена программная реализация алгоритма адаптивной маршрутизации, позволяющая повысить эффективность процессов маршрутизации и уменьшить затраты на проектирование маршрутов передачи данных в корпоративных сетях нескольких провайдеров связи с различными зонами покрытия абонентов.

Ключевые слова: *корпоративные сети, алгоритмы маршрутизации, адаптивная маршрутизация, провайдеры связи, зона покрытия, абоненты*

Введение

В настоящее время операторы связи сталкиваются с растущей сложностью телекоммуникационных сетей, ростом объема передаваемой информации и увеличением числа устройств, подключенных к операторским сетям. Активное распространение сетевых устройств заставляет провайдеров связи искать возможности быстрого снижения текущих расходов при непрерывном внедрении новых услуг и технологий. Возникают задача оптимизации структуры сети по критерию минимальной стоимости в кратчайшие сроки и в реальном времени, а также задача поиска оптимальных маршрутов в уже построенной ранее сети. Для решения перечисленных задач обычно используют алгоритмы построения минимальных покрывающих деревьев, а также алгоритмы адаптивной маршрутизации. Наиболее часто используют классические алгоритмы Прима, Дейкстры, Беллмана—Форда. Однако в реальности сетевая структура усложняется наличием нескольких провайдеров связи, которые предоставляют альтернативные каналы связи для поиска оптимальных маршрутов, а также динамическими изменениями параметров сети. Классические алгоритмы не учитывают данные факторы, поэтому необходимо разработать новые эффективные алгоритмы, позволяющие решать задачи маршрутизации в различных сетевых структурах.

В соответствии с исследованием Digital Consumer Survey [1], представленным компанией Accenture, для управления коммуникационными и развлекательными продуктами потребители используют, как правило, только одного поставщика услуг. По итогам работы конгресса [2] установлено, что для обеспечения потребительских предпочтений необходимо одновременно использовать услуги нескольких поставщиков связи.

Согласованная работа провайдеров связи, объединение знаний в строительстве и управлении IP-сетями, совместное использование возможностей для разработки и предоставления новых, ориентированных на клиента услуг могут стать особым преимуществом в работе операторов и позволить снизить расходы на предоставление услуг связи. Сотрудничество провайдеров позволит создать новые источники дохода, повысить качество и ускорить развитие услуг связи, увеличить скорость передачи данных и число привлеченных клиентов, а также создать инновационные сетевые технологии.

В работах [3—6] предложены новые методы и подходы к маршрутизации, основанные на использовании программно-конфигурируемых сетей, на механизмах сотрудничества провайдеров связи, переносе сетевой структуры в "облако", на разработке программируемого сетевого оборудования. В работах [7—10] предложены идеи формирования структуры корпоративной сети при наличии альтернативных каналов связи между базовыми узлами. Однако вопросы дальнейшей маршрутизации трафика в отмеченных работах не рассматривались. В связи с этим обстоятельством разработка алгоритмов адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи с различными зонами покрытия абонентов является актуальным направлением. Применение новых перспективных подходов для решения задачи адаптивной маршру-

тизации позволит снизить стоимость построения и обслуживания корпоративных сетей нескольких провайдеров связи с различными зонами покрытия.

Постановка задачи

В общем случае корпоративную сеть представляют в виде множества базовых (коммуникационных) узлов, соединенных скоростными каналами связи, к которым подключают пользователей сети. Стоимость маршрута складывается из стоимостей используемых каналов связи и базовых узлов.

Для решения задачи поиска оптимальных маршрутов в корпоративных сетях широко используют алгоритм Дейкстры. Этот алгоритм применяется для построения таблиц маршрутизации в протоколе OSPF (*Open Shortest Path First*). Характеристики, метрики и параметры качества обслуживания протокола OSPF подробно рассмотрены в работах [11—13]. Трудоемкость построения таблиц маршрутизации с использованием классического алгоритма Дейкстры оценивают как $O(N^2)$, где N — число маршрутизаторов корпоративной сети.

Однако на практике задача поиска оптимальных маршрутов в корпоративной сети осложняется наличием нескольких альтернативных вариантов ее решения. Это обстоятельство приводит к необходимости стыковки каналов в узлах сети и, как следствие, к зависимости стоимости обслуживания узла от подключаемых к нему каналов связи. Примером может служить решение задачи оптимального построения базовой сети региона при наличии нескольких операторов связи с различными зонами покрытия [9, 10]. Трудоемкость построения таблиц маршрутизации корпоративных сетей нескольких провайдеров связи с применением классического алгоритма Дейкстры составляет значение порядка $O(N^3)$ за счет добавления цикла по всем узлам для определения наименьшего канала связи между любой парой узлов сети.

В общем случае для решения данной задачи применяется графовая модель корпоративной сети, у которой вершины графа — базовые узлы (маршрутизаторы, коммутаторы), ребра графа — линии связи. На практике весу ребра могут соответствовать стоимость аренды канала связи, затраты на оплату единицы трафика, передаваемого по каналу связи, соответствующему данному ребру, либо более сложная функция, учитывающая большее число параметров корпоративной сети. При функционировании в корпоративной сети нескольких провайдеров связи между отдельными базовыми узлами возможно наличие нескольких каналов связи, что соответствует нескольким ребрам, связывающим соответствующие вершины графа.

Разработка новых, более эффективных алгоритмов адаптивной маршрутизации позволяет: оптимизировать структуры корпоративных сетей нескольких провайдеров связи; снизить стоимость построения и обслуживания сетей; повысить скорость передачи данных по оптимальным маршрутам в сети.

Разработка алгоритма

Для повышения эффективности функционирования корпоративных сетей нескольких провайдеров связи с различными зонами покрытия абонентов авторы предлагают:

1) использовать математическую модель, алгоритм и программное обеспечение, позволяющие уменьшить трудоемкость построения таблиц маршрутизации при динамических изменениях нагрузки на линиях связи;

2) не проводить полный пересчет маршрутных таблиц, что, как правило, предлагается в известных алгоритмах.

Математическую модель корпоративной сети нескольких провайдеров связи представим в виде мультиграфа $G = (V, E, W, Z, S)$, где V — множество вершин (узлов связи или маршрутизаторов); $|V| = N$; E — множество ребер (каналов или линий связи); $|E| = M$; W — множество весов ребер (стоимость каналов связи между узлами); Z — множество провайдеров связи, $|Z| = m$; S — множество весов вершин (стоимость подключения каналов связи к базовым узлам), $|S| = K$.

Каждая вершина v_i графа G также имеет свой вес s_i. Вес вершины может определяться, например, числом пользователей, которые подключаются к узлу, которому соответствует данная вершина. Однако в случае подключения к узлу каналов связи нескольких провайдеров он выполняет функции по коммутации этих каналов, что, как правило, требует дополнительных затрат. Поэтому вес вершины должен учитывать этот фактор, что в нашем случае будет выглядеть следующим образом:

$$s_i = \sum_{i=1}^{m} s_j^i,$$

где s_j^i — вес вершины j, связанный с обслуживанием каналов провайдера связи с номером i ($i = 1...m$, $j = 1...N$, где m — число провайдеров связи; N — число вершин).

Для разработки алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи необходимо ввести временные метки по каждому провайдеру и уменьшать их в соответствии с классическим алгоритмом Дейкстры. Отличие будет заключаться в вычислении постоянной метки вершины

$$\min_{i=1..m} \left\{ d_j^i \right\},$$

где d_j^i — метка вершины с номером j по провайдеру связи с номером i.

На основе предложенной модели сети и изложенных выше положений разработан алгоритм адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи. Рассмотрим работу данного алгоритма по шагам.

Шаг 1. Присвоение начальных значений. Положить $d_b^i = s_b^i$ и считать эту метку постоянной; v_b — начальная вершина, v_p — текущая вершина. Принять $d_j^i = \infty$ для всех $v_j \neq v_b$ и считать эти метки временными. Присвоить $p = b$.

Шаг 2. Обновление меток. Для всех вершин $v_j \in G(v_p)$, инцидентных вершине v_p и имеющих временные метки, изменить метки в соответствии с выражением:

$$d_j^i = \min_{i=1...m}\left(d_j^i, \min_{i=1...m}\left\{d_p^i\right\} + w_{p,j}^i + s_p^i + s_j^i\right), \quad (1)$$

если d_p^i не включает стоимость s_p^i i-го провайдера связи, или по выражению

$$d_j^i = \min_{i=1...m}\left(d_j^i, \min_{i=1...m}\left\{d_p^i\right\} + w_{p,j}^i + s_j^i\right), \quad (2)$$

если d_p^i уже включает стоимость s_p^i i-го провайдера связи, где d_p^i — метка текущей вершины; d_j^i — метка вершины, смежной с текущей вершиной; $w_{p,j}^i$ — вес ребра, соединяющего вершину v_p с вершиной v_j.

Шаг 3. Превращение метки d_j^i в постоянную d_j^*. Среди всех вершин с временными метками найти такую, для которой

$$d_j^* = \min_{i=1...m}\left\{d_j^i\right\}. \quad (3)$$

Шаг 4. Считать метку d_j^* постоянной и положить $v_p = v_j^*$, где v_j^* — вершина с постоянной меткой d_j^*.

Шаг 5. Если все вершины имеют постоянные метки, то конец алгоритма. Иначе перейти к шагу 2.

Сам маршрут можно найти, применяя рекурсивно процедуру, в которой реализовано выражение

$$d_j^* = \min_{i=1...m}\left\{d_j^{i*}\right\} + w_{j,k}^i + s_p^i + s_j^i. \quad (4)$$

Оно справедливо для последующей и предыдущей вершин, принадлежащих одному маршруту. Кроме того, если узел связи имеет каналы нескольких провайдеров связи, то необходимо изменить постоянную метку d_j^* следующим образом:

$$d_j^{**} = d_j^* + s_j^i,$$

где d_j^{**} — метка вершины v_j^* с учетом веса этой вершины, связанного с обслуживанием каналов провайдера связи с номером i.

Программная реализация алгоритма

Для подтверждения правильности предложенного алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи с различными зонами покрытия абонентов разработано программное обеспечение моделирования процессов адаптивной маршрутизации. Основное внимание при разработке уделялось корректности предлагаемого алгоритма и размерности решаемой задачи. Средствами разработки программного обеспечения моделирования корпоративных сетей и процессов маршрутизации выбраны среда визуального программирования Microsoft Visual Studio, платформа Microsoft.Net 4.0, язык программирования C#. Такой выбор обусловлен следующими факторами:

- удобная визуальная среда программирования;
- высокая скорость работы компилятора;
- высокое быстродействие откомпилированных программ;
- широкий выбор функций и библиотек для программирования графической части приложения;
- большое число визуальных компонентов для создания пользовательского интерфейса.

Разработанное программное обеспечение предназначено для выполнения следующих функций:

- создание и визуальное редактирование графовых моделей корпоративных сетей нескольких провайдеров связи;
- поиск оптимальных маршрутов на графе с использованием алгоритмов адаптивной маршрутизации;
- проведение экспериментов по изменению весов ребер графа, добавлению и удалению вершин и ребер графа с построением дерева кратчайших путей;
- оценка трудоемкости алгоритмов адаптивной маршрутизации.

Интерфейс программной системы приведен на рис. 1 (см. вторую сторону обложки).

Для каждого узла сети можно задать максимальный исходящий трафик в мегабайтах, цену за мега-

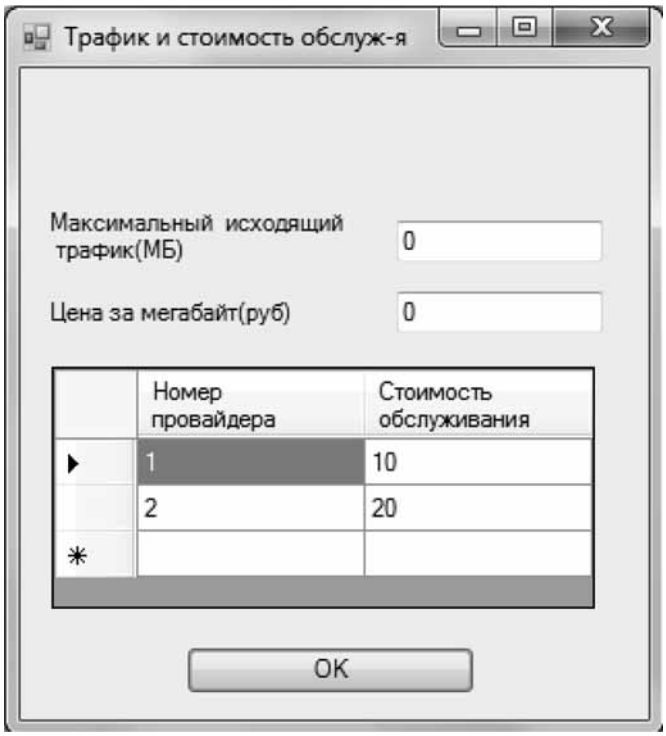


Рис. 2. Настройки параметров узла связи

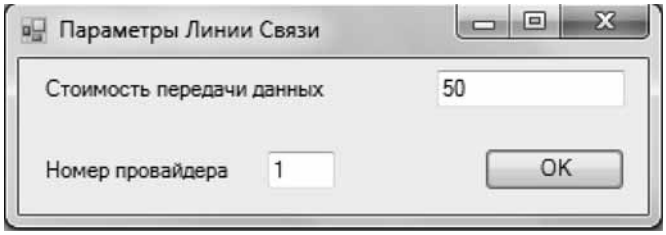


Рис. 3. Настройка параметров линии связи

байт переданных данных, стоимость обслуживания по каждому провайдеру связи. Настройки параметров узла связи приведены на рис. 2.

Для каждой линии связи задаются номер обслуживающего провайдера и стоимость передачи данных по этому каналу связи. Настройки параметров линии связи приведены на рис. 3.

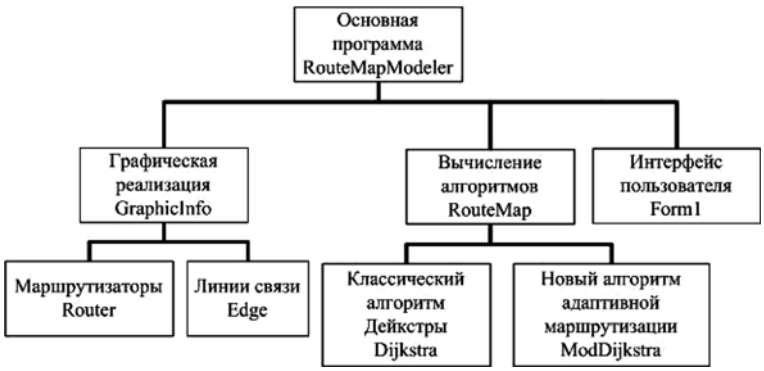


Рис. 4. Компоненты программного обеспечения

Разработанное программное обеспечение Route MapModeler состоит из следующих основных классов:

- RouteMap — содержит методы вычисления алгоритмов адаптивной маршрутизации;
- GraphicInfo — содержит графическую реализацию алгоритмов и состоит из следующих элементов:
  - ♦ Router — реализует графическое представление и параметры маршрутизаторов;
  - ♦ Edge — реализует графическое представление и параметры линий связи.

На рис. 4 укрупненно представлены компоненты предлагаемого программного обеспечения.

Основным вычислительным методом в классе RouteMap является метод реализации предложенного алгоритма маршрутизации в корпоративных сетях нескольких провайдеров связи — метод public void ModDijkstra(). Переменная *countRouters* обозначает число маршрутизаторов в сети, *providersCount* — число провайдеров сети, *current* — номер текущей вершины, *Graph* [*i*, *j*][*k*] — список смежности весов ребер между вершинами *i* и *j* по провайдеру *k*, *routers*[*i*]. *dMark*[j] — временная метка маршрутизатора *i* по провайдеру *k*, *routers*[*i*].*looked* — признак просмотренной вершины *i* (маршрутизатора), массив *u*[*i*] — массив непросмотренных вершин, *routers*[*j*].*weightByProvider*[*k*] — вес маршрутизатора *j*, обслуживаемого провайдером *k*.

На этапе присвоения начальных значений считается, что начальная вершина *routers*[0] уже просмотрена и ее метка устанавливается постоянной. Фрагмент кода присвоения начальных значений имеет следующий вид:

```
for (int k = 0; k < providersCount; k++)
 {
    routers[0].dMark[k] = 0;
 }
routers[0].looked = true;
u[0] = true;
current = 0;
```

Выбор новой текущей вершины и превращение временной метки в постоянную метку реализованы следующим фрагментом кода:

```
double minMark = INF;
foreach (Router router in routers)
{
      if (router.looked == false)
      {
          foreach (double mark in router.dMark)
          {
              if (mark < minMark)
              {
                  minMark = mark;
                  current = router.Rnumber;
              }
          }
      }
}
routers[current].looked = true;
u[current] = true;
```

Далее в соответствии с приведенным выше алгоритмом происходит обновление меток вершин. Для всех вершин, инцидентных текущей вершине и имеющих временные метки, изменяются метки в соответствии с выражением (1), т. е. если метка текущей вершины не включает стоимость обслуживания *k*-го провайдера связи, то используем следующий фрагмент кода:

```
for (int j = 0; j < routers.Count; ++j)
{
   for (int k = 0; k < providersCount; k++)
   {
      if (graph[a, j][k] < INF)
      {
         if (!u[j])
         {
           routers[j].dMark[k] = Math.Min(routers[j].dMark[k],
           routers[a].dMark.Min() + graph[a, j][k] +
           routers[a].weightByProvider[k] + routers[j].weightByProvider[k]);
         }
      }
```

или, в соответствии с выражением (2), если метка текущей вершины уже включает стоимость обслуживания *k*-го провайдера связи, используется следующий код:

```
for (int j = 0; j < routers.Count; ++j)
{
    for (int k = 0; k < providersCount; k++)
    {
       if (graph[a, j][k] < INF)
       {
         if (!u[j])
         {
           routers[j].dMark[k] = Math.Min(routers[j].dMark[k],
            routers[a].dMark.Min() + graph[a, j][k]+
             routers[a].weightByProvider[k]);
         }
       }
```

После пересчета всех меток проводится поиск следующего канала связи, который войдет в оптимальный маршрут:

```
Edge findingEdge = null;
foreach (Edge ed in edges)
{
   if (((ed.a.Rnumber == a) && (ed.b.Rnumber == j) || (ed.b.Rnumber == a) &&
(ed.a.Rnumber == j)) && (ed.ProvNumber == k))
   {
     findingEdge = ed;
     break;
   }
}
foreach (PathContainer pc in routers[j].alternativeRouteMark)
{
   if (pc.edge! = findingEdge)
   {
     routers[j].alternativeRouteMark.Add(new PathContainer(findingEdge,
     routers[a].dMark.Min() + graph[a, j][k] +
     routers[a].weightByProvider[k] + routers[j].weightByProvider[k]));
   }
}
```

```
public struct PathContainer
{
      public Edge edge;
      public double rating;
      public PathContainer(Edge e, double r)
{
        edge = e;
        rating = r;
    }
}
```

Если все роутеры были просмотрены, то алгоритм завершает свою работу, в противном случае снова выбирается новая текущая вершина и выполняется еще одна итерация алгоритма.

В результате выполнения всех шагов алгоритма дерево оптимальных маршрутов будет иметь вид, представленный на рис. 5 (см. вторую сторону обложки).

В рамках тестовой апробации предложенного алгоритма адаптивной маршрутизации были проведены численные расчеты графовых моделей корпоративных сетей, состоящих из 10, 50 и 100 вершин с учетом двух провайдеров связи в сети.

В таблице приведено сравнение классического алгоритма Дейкстры и предложенного алгоритма в корпоративных сетях нескольких провайдеров связи. Стоимости маршрутов между узлом связи Router 1 и остальными узлами сети в соответствии с протоколом OSPF вычислялись следующим образом:

Стоимость = 100 (Мбит/с) / полоса пропускания канала связи (Мбит/с).

Исследование разработанного алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи показало, что трудоемкость построения оптимальных маршрутов передачи данных сопоставима с алгоритмом Дейкстры и составляет значение порядка $O(N^3)$. Однако предложенный алгоритм адаптивной маршрутизации позволяет уменьшить стоимость аренды каналов связи и затраты на оплату единицы трафика по каждому из маршрутов за счет исключения из стоимости маршрутов стоимости неиспользуемых каналов и узлов связи. Разработанное программное обеспечение подтверждает эффективность предложенного алгоритма построения оптимальных маршрутов передачи данных в корпоративных сетях нескольких провайдеров связи.

Сравнительные характеристики алгоритмов

Параметры		Алгоритм	
		классический	новый
Трудоемкость		$O(N^3)$	$O(N^3)$
Время работы алгоритма, мс		5,82	5,76
Стоимость маршрута в соответствии с протоколом OSPF	Router 1 — Router 2	140	130
	Router 1 — Router 3	150	110
	Router 1 — Router 4	130	130
	Router 1 — Router 5	240	210
	Router 1 — Router 6	330	290
	Router 1 — Router 7	290	230
	Router 1 — Router 8	370	320

Заключение

Разработанный алгоритм и программное обеспечение адаптивной маршрутизации позволяют снизить стоимость построения структур корпоративных сетей нескольких провайдеров связи с различными зонами покрытия абонентов за счет оптимального построения маршрутов передачи данных и исключения из их стоимости стоимости неиспользуемых каналов и узлов связи.

Работа выполнена при финансовой поддержке гранта Президента РФ для молодых ученых — кандидатов наук МК-819.2014.9.

Список литературы

1. **Управление** IT-услугами. URL: https://www.accenture.com/us-en/insight-consumer-banking-survey.aspx (дата доступа 01.07.2015).

2. **Mobile** World Congress 2015, Барселона. URL: http://www.mobileworldcongress.com (дата доступа 01.07.2015).

3. **Jeong-Woo Cho, Yung Yi.** On the Payoff Mechanisms in Peer-Assisted Services with Multiple Content Providers: Rationality and Fairness // Networking, IEEE/ACM Transactions. 2013. Vol. 22, Issue 3. P. 731—744.

4. **Sha Hua, Pei Liu, Shivendra S. Panwar.** The Urge to Merge: When Cellular Service Providers Pool Capacity // Communications (ICC), IEEE International Conference. 2012. P. 5020—5025.

5. **Dong Yuan, Xiao Liu, Lizhen Cui** et al. An Algorithm for Cost-Effectively Storing Scientific Datasets with Multiple Service Providers in the Cloud // eScience, IEEE 9th International Conference. 2013. P. 285—292.

6. **McKeown N., Anderson T., Balakrishnan H.** et al. Openflow: Enabling Innovation in Campus Networks // ACM SIGCOMM Computer Communication Review. 2008. Vol. 38, N 2. P. 69—74.

7. **Бурков С. М.** Алгоритмы и методы поэтапного формирования телекоммуникационных сетей региона. Математическая модель // Вестник Тихоокеанского государственного университета. 2008. № 1. С. 91—100.

8. **Бурков С. М., Бертенев В. А.** Концептуальный подход к созданию телекоммуникационных систем с поэтапным развитием // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. 2009. Т. 4, № 82. С. 7—15.

9. **Бурков С. М., Бертенев В. А.** Постановка задачи формирования базовой сети регионального уровня // Научно-технические ведомости Санкт-Петербургского государственного

политехнического университета. Информатика. Телекоммуникации. Управление. 2009. Т. 4, № 82. С. 22—27.
10. **Горшков С. Г., Никитин Е. В., Саксонов Е. А.** Задача формирования структуры базовой сети // Вестник Тихоокеанского государственного университета. 2010. № 2. С. 59—66.
11. **Перепелкин Д. А.** Алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов корпоративной сети // Вестник Рязанского государственного радиотехнического университета. 2010. № 34. С. 65—71.
12. **Корячко В. П., Перепелкин Д. А.** Корпоративные сети: технологии, протоколы, алгоритмы. М.: Горячая линия — Телеком, 2011. 219 с.
13. **Корячко В. П., Перепелкин Д. А.** Анализ и проектирование маршрутов передачи данных в корпоративных сетях. М.: Горячая линия — Телеком, 2012. 235 с.

D. A. Perepelkin, Associate Professor, e-mail: dmitryperepelkin@mail.ru, **M. A. Ivanchikova,** Student, Ryazan State Radio Engineering University

Software Development of Adaptive Routing Processes Modeling in Corporate Networks of Multiple Service Providers with Different Covering Areas of Subscribers

At the present day service providers are faced with the growing complexity of telecommunications networks, growth in the volume of data transmitted and increasing the number of devices connected to the operating networks. Active distribution of network devices makes communications providers find opportunities to rapidly reduce the current costs with continuous introduction of new services and technologies. The problem of optimizing the structure of the network by the criterion of minimum cost in the shortest possible time and in real time, as well as the task of finding the optimal routes has previously constructed a network is arises. Coordinated work of service providers, combining expertise in building and managing IP-networks, sharing opportunities for the development and delivery of new customer-oriented services can be particularly advantageous in the work of operators will reduce cost of providing telecommunications services. Collaboration of providers create new revenue streams, improve the quality and accelerate the development of telecommunications services, increase the data rate and the number of involved customers, as well as create innovative networking technology.

In connection with this, the development of adaptive routing algorithms in corporate networks of multiple service providers with different covering areas of subscribers are the actual direction.

Mathematical model and adaptive routing algorithm which reduce the complexity of building a routing table in the course of dynamic changes of the load on the communication lines, as well as produce a complete recalculation of the routing tables are offered in article.

Software of adaptive routing processes modeling, confirming the validity of the proposed model and adaptive routing algorithm in corporate networks of multiple service providers with different covering areas of subscribers is developed. As part of the validation test it was carried out numerical calculations graph models of corporate networks, consisting of 10, 50 and 100 communication nodes in the presence of two service providers in the network. The research of the adaptive routing algorithm show its effectiveness in comparison with existing analogues by constructing an optimal routes data and exclusion of their cost — the cost of unused channels and communication nodes.

Keywords: *corporate networks, routing algorithms, adaptive routing, communication providers, covering area, subscribers*

References

1. **IT-service** management, available at: https://www.accenture.com/us-en/insight-consumer-banking-survey.aspx (data of access 01.07.2015).
2. **Mobile** World Congress 2015, Barselona, available at: http://www.mobileworldcongress.com (data of access 01.07.2015).
3. **Jeong-Woo Cho, Yung Yi.** On the Payoff Mechanisms in Peer-Assisted Services with Multiple Content Providers: Rationality and Fairness. *Networking, IEEE/ACM Transactions.* 2013, vol. 22, issue 3, P. 731—744.
4. **Sha Hua, Pei Liu, Shivendra S. Panwar.** The Urge to Merge: When Cellular Service Providers Pool Capacity. *Communications (ICC), IEEE International Conference*, 2012, pp. 5020—5025.
5. **Dong Yuan, Xiao Liu, Lizhen Cui, Tiantian Zhang, Wenhao Li, Dahai Cao, Yun Yang.** An Algorithm for Cost-Effectively Storing Scientific Datasets with Multiple Service Providers in the Cloud. *eScience, IEEE 9th International Conference*, 2013, pp. 285—292.
6. **McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J.** Openflow: Enabling Innovation in Campus Networks // ACM SIGCOMM Computer Communication Review, 2008, vol. 38, no. 2, pp. 69—74.
7. **Burkov S. M.** Algoritmy i metody pojetapnogo formirovanija telekommunikacionnyh setej regiona. Matematicheskaja model'. *Vestnik Tihookeanskogo gosudarstvennogo universiteta*, 2008, vol. 1, pp. 91—100 (in Russian).
8. **Burkov S. M., Bertenev V. A.** Konceptual'nyj podhod k sozdaniju telekommunikacionnyh sistem s pojetapnym razvitiem. *Nauchno-tehnicheskie Vedomosti Sankt-Peterburgskogo Gosudarstvennogo Politehnicheskogo Universiteta. Informatika. Telekommunikacii. Upravlenie*, 2009, vol. 82, pp. 7—15 (in Russian).
9. **Burkov S. M., Bertenev V. A.** Postanovka zadachi formirovanija bazovoj seti regional'nogo urovnja. *Nauchno-tehnicheskie Vedomosti Sankt-Peterburgskogo Gosudarstvennogo Politehnicheskogo Universiteta. Informatika. Telekommunikacii. Upravlenie*, 2009, vol. 82, pp. 22—27 (in Russian).
10. **Gorshkov S. G., Nikitin E. V., Saksonov E. A.** Zadacha formirovanija struktury bazovoj seti. *Vestnik Tihookeanskogo Gosudarstvennogo Universiteta*, 2010, vol. 2, pp. 59—66 (in Russian).
11. **Perepelkin D. A.** Algoritm adaptivnoj uskorennoj marshrutizacii na baze protokola OSPF pri dinamicheskom dobavlenii jelementov korporativnoj seti. *Vestnik Rjazanskogo Gosudarstvennogo Radiotehnicheskogo Universiteta*, 2010, vol. 34, p. 65—71 (in Russian).
12. **Korjachko V. P., Perepelkin D. A.** *Korporativnye seti: tehnologii, protokoly, algoritmy* (Corporate Networks: Technologies, Protocols, Algorithms), Moscow, Gorjachaja linija — Telekom, 2011, 219 p. (in Russian).
13. **Korjachko V. P., Perepelkin D. A.** *Analiz i proektirovanie marshrutov peredachi dannyh v korporativnyh setjah* (Analysis and Design of Data Transmission Routes in Corporate Networks), Moscow, Gorjachaja linija — Telekom, 2012, 235 p. (in Russian).

УДК 004.89

П. Д. Блинов, инженер-программист, e-mail: blinoff.pavel@gmail.com,
Вятский государственный гуманитарный университет, г. Киров

Программная система аспектно-эмоционального анализа текста

Представлено решение задачи автоматического анализа тональности текста. Рассмотрен наиболее детальный вариант такого анализа, когда выраженные мнения определяются сразу для нескольких аспектов одного объекта. Предложена структура системы анализа мнений, приведен состав подсистем и их назначение. С помощью диаграммы классов показан один из вариантов программной реализации системы. Эксперименты, проведенные на основе общедоступных текстовых корпусов, подтвердили эффективность разработанной системы.

Ключевые слова: *обработка естественного языка, автоматический анализ тональности, аспектно-эмоциональный анализ текста, извлечение мнений, диаграмма классов, программная система*

Введение

Многие текстовые документы, представленные пользователями в сети Интернет, являются источником субъективной и оценочной информации, т. е. содержат определенные мнения ее авторов. *Мнением* называют эмоциональное отношение автора, выраженное в тексте относительно объекта высказывания, в качестве которого может выступать материальная сущность или ее атрибуты, событие или процесс [1]. На основе обобщения мнений многих пользователей относительно одного объекта представляется возможность извлечь новые знания об этом объекте [2]. Например, из коллекции текстовых отзывов пользователей о конкретном ресторане могут быть получены сведения об уровне цен, качестве обслуживания и кухни, интерьере заведения. Задачу автоматического выявления мнений в тексте называют *аспектно-эмоциональным анализом* (*aspect-based sentiment analysis*) или *извлечением мнений* (*opinion mining*) [1]. Название указывает на то, что мнения определяются для конкретных аспектов. Под *аспектом* или *аспектной категорией* подразумевается конкретное свойство, составная часть или характеристика исследуемого объекта. В примере о ресторане такие аспекты — это *цена*, *кухня*, *сервис* и *интерьер*. Такой аспектно-эмоциональный анализ, как правило, сводится к решению следующих основных подзадач:

1) извлечение аспектных терминов, т. е. выявление выражений, которыми обозначают конкретные аспекты (например, аспект, именуемый сервисом, может определяться такими выражениями, как *официант*, *девушка-официантка*, *обслуживание*, *сотрудник*, *персонал*, *администратор* и др.);

2) определение тональности выраженной по отношению к извлеченным аспектным терминам (например, в предложении "***Официант*** *был нерасторопный и грубый*" термин ***официант***, принадлежащий аспекту *сервис*, упомянут с негативной тональностью);

3) вычисление интегральных оценок тональности по каждой из аспектных категорий (например, на основании проведенного анализа конкретному отзыву о ресторане могут быть присвоены следующие оценки: *кухня* — позитивная; *сервис* — негативная и т. д.).

Следует отметить, что решения перечисленных задач осложняются рядом особенностей. Например, сильной контекстной зависимостью выражений тональности, лексической вариативностью аспектных терминов и др.

Проблема автоматического анализа тональности активно исследуется с начала 2000-х гг. Ее решение может быть выполнено в рамках двух подходов [1] на основе:

1) знаний (*knowledge-based approach*);

2) машинного обучения (*machine learning approach*).

Первый подход для определения мнений опирается на шаблоны и правила, составленные экспертами предметной области [3, 4]. Тщательно сформированный набор правил позволяет добиваться высокой точности анализа. Основной недостаток подхода заключается в трудоемкости составления правил. Также затруднительно повторное использование существующих правил при перенастройке системы на другие предметные области. Представителями систем, реализующих этот подход для анализа русского языка, являются RCO Fact Extractor [5] и Аналитический курьер [6].

Подход на основе машинного обучения предполагает автоматическое построение алгоритма для выявления мнений из текста [7, 8], что позволяет избавиться от рутинного этапа составления правил. Однако эффективность работы систем, реализованных в рамках этого подхода, во многом зависит от количества и качества обучающих данных. По этой

причине реализация и использование таких программных систем остается прерогативой крупных компаний, обладающих необходимым объемом данных. Примером такой системы для анализа английского языка является AlchemyAPI [9] от одноименной компании.

В настоящей работе предложены структура системы аспектно-эмоционального анализа текста и вариант ее программной реализации. Методы, реализованные в системе и описанные в работах [10, 11], выполняют извлечение мнений в рамках подхода на основе машинного обучения без учителя, используя семантические свойства пространства распределенных представлений слов [12]. При скромном объеме экспертных знаний система позволяет на качественном уровне выявлять мнения, что доказано рядом экспериментов на англоязычных [13] и русскоязычных корпусах документов [10, 14].

Общая структура системы

Система аспектно-эмоционального анализа состоит из нескольких подсистем, схема взаимодействия которых показана на рис. 1. Как и в традиционных системах обработки текста на основе машинного обучения с учителем, на схеме сохранено деление текстов на **обучающий** и **тестовый** корпуса. Однако в предлагаемой системе обучающий корпус, в отличие от традиционного его понимания, может не содержать никакой экспертной разметки. Его роль заключается в предоставлении текстов заданной предметной области, содержащих статистику совместной встречаемости терминов. Такая статистическая информация используется в дальнейшем при построении пространства распределенных представлений слов.

Все тексты, с которыми работает система, подвергаются **предобработке**. В соответствующей подсистеме происходит разбиение текста на токены и выполнение морфологического анализа. Эта же подсистема содержит блок, позволяющий учитывать лексическую структуру выражений. В этом блоке выполняется обработка отрицаний для учета возможной инверсии выражений тональности. Например, последовательность токенов "*второй зал не очень понравился*" становится "*второй зал не_очень_понравился*". Также этот блок позволяет организовать учет дополнительных лексических структур, например, выделение словосочетаний или устойчивых оборотов речи. После такой обработки каждый текст представляется последовательностью простых и составных лексических единиц.

Подсистема представления слов реализует функции по построению пространства распределенных представлений слов. Подсистема работает на подго-

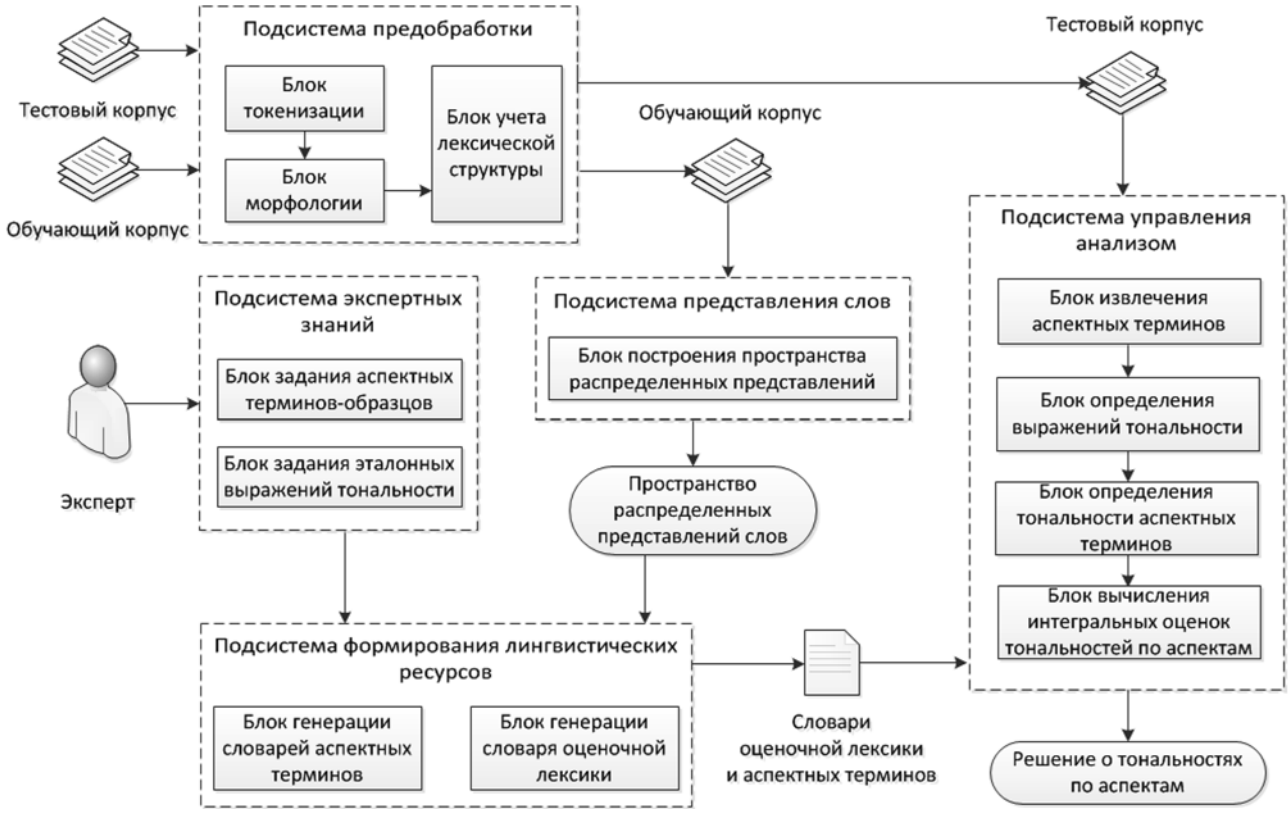


Рис. 1. Структура системы аспектно-эмоционального анализа

товленном корпусе обучающих документов, а ее результатом является соответствие лексических единиц векторам пространства распределенных представлений [10]. Распределенным представлением слова (*distributed representation of word*) называют d-мерный вектор признаков $\mathbf{a} = (a_1, ..., a_d)$, $a_i \in R$, где R — множество вещественных чисел. Общая идея таких представлений заключается в том, чтобы закодировать в компонентах вектора признаки, обнаруживающие значение термина через его контекст. Например, термины *фильм* и *кино* имеют схожее значение. Как следствие, и контексты, в которых они встречаются, тоже похожи (*смотрели это кино*, *фильм досмотрели, отличное кино*, *отличный получился фильм* и др.). Поэтому и векторы этих терминов будут близки исходя из определенной меры близости, например меры косинусного сходства [15]. Для построения такого пространства подсистема использует модель Skip-gram [12].

Подсистема экспертных знаний выполняет роль интерфейса между экспертом и системой. Эксперт определяет разбиение целевого объекта на аспектные категории, а также задает небольшой набор начальных терминов-образцов, характеризующих каждый из аспектов. Например, аспект *кухня* может быть задан следующим набором терминов: *кухня*, *блюдо*, *хлеб*, *суп*, *десерт*. Аналогично эксперт определяет эталонные выражения тональности для данной предметной области. Например, для ресторанной тематики позитивная тональность может определяться следующими выражениями: *вкусно*, *уютная*, *недорого*, *быстро*, *отличный*; а негативная тональность может определяться выражениями: *хамить*, *накурено*, *подгорелый*, *черствый*, *ужасный*.

Подсистема формирования лингвистических ресурсов работает на основе полученного ранее пространства распределенных представлений слов и экспертных знаний. Для каждого из заданных экспертом аспектов генерируются аспектные термины [10]. Отбор терминов проводится итеративным способом. На каждой итерации определяется набор лексических единиц, наиболее близких (согласно косинусной мере) к терминам, заданным экспертом. После нескольких таких итераций для каждой аспектной категории формируются словари терминов.

Такой метод отбора терминов обладает следующим преимуществом — возможностью обнаружения лексем с распространенными ошибками в написании, например, *интерьер — интерьр*, *официантка — официатка*, *хачапури — хачипури* и др. Эта возможность появляется благодаря свойствам пространства распределенных представлений, в котором слова с ошибками оказываются близки в смысле косинусного расстояния к словам без ошибок. При этом контекст каждого слова оказывается более устойчивым к ошибкам написания, т. е. если термин написан с ошибкой, он появляется в том же контексте, что и правильно написанный термин, поэтому становится возможным его извлечение.

Этой подсистемой также создается словарь оценочной лексики. Построение словаря выполняется в два отмеченных далее этапа [11].

1. Отбор кандидатов в эмоциональные выражения. Прежде всего отбираются прилагательные и глаголы, поскольку слова этих частей речи несут наибольшую эмоциональную нагрузку. Также выполняется извлечение сложных выражений, содержащих модификаторы усиления (*сильно*, *очень*, *совсем* и др.) и отрицания (в основном, частица *не*). Как уже было отмечено выше, учет и создание таких составных выражений выполняются подсистемой **предобработки**.

2. Взвешивание полученных выражений для получения оценки их тональности. На этом этапе проводится вычисление числовых оценок для всех кандидатов, отобранных на первом этапе. Прежде всего, вычисляют оценку сходства кандидата с эталонными выражениями тональности, заданными в **подсистеме экспертных знаний**. При наличии обучающих данных вычисляются дополнительные оценки на основе метода взаимной информации PMI (*Pointwise Mutual Information*) [16]:

$$score(w) = PMI(w, pos) - PMI(w, neg),$$

где $PMI(w, s)$ — оценка взаимной информации между кандидатом w и классом тональности s, $s \in \{pos, neg\}$ (*pos* — позитивная тональность, *neg* — негативная тональность).

После этого этапа каждый из кандидатов в эмоциональные выражения ассоциирован с двумя оценками. Одновременное вычисление двух оценок позволяет частично избегать ошибок в определении значений выражений тональности. Например, тональности следующих выражений при использовании лишь одной оценки были бы определены неверно: *выгодный* −0,7/+2,5; *суховатый* +1,4/−1,9; *зажимать* +0,2/−2,4; *горчить* +0,6/−2,7; *не цеплять* +0,4/−2,6 и т. д. В этих примерах первая оценка — оценка косинусного сходства с эталонными выражениями тональности, а вторая оценка — оценка, вычисленная на основе метода взаимной информации. Для выражения с негативной тональностью — *горчить* — первая оценка скорее неверна (+ 0,6), однако вторая оценка правильна (−2,7) и больше по абсолютному значению, поэтому итоговая тональность этого выражения будет определена верно.

Полученные словари используются **подсистемой управления анализом** для извлечения из тестовых данных аспектных терминов, для определения выражений тональности и вычисления общих оценок тональностей документа в соответствии с аспектными категориями, определенными экспертом.

Программная реализация системы

Проектирование программной системы выполнено на основе принципов объектно-ориентированного программирования с помощью языка моделирования UML [17]. Ядро системы представлено классами, изображенными на рис. 2. Некоторые служебные и вспомогательные классы, например класс

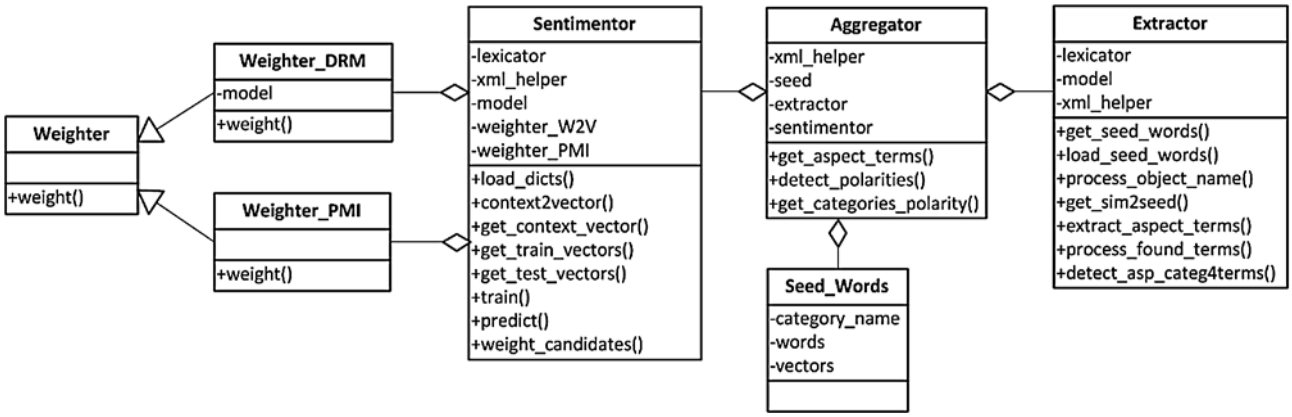


Рис. 2. Диаграмма основных классов системы

`Preprocessor`, реализующий подсистему предобработки, не представлены на диаграмме. Для простоты оставлены лишь ключевые методы классов, сигнатуры методов опущены.

Между классами можно выделить два основных вида отношений:

- стрелка с треугольным окончанием представляет отношение наследования, также называемое отношением обобщения (например, класс `Weighter _ PMI` является наследником класса `Weighter`);
- стрелка с ромбом представляет отношение агрегации (например, класс `Sentimentor` включает в себя (агрегирует) классы `Weighter _ PMI` и `Weighter _ DRM`).

Функции подсистем формирования лингвистических ресурсов и управления анализом реализованы в основных классах `Extractor`, `Sentimentor` и `Aggregator`.

Класс `Aggregator` управляет всем процессом обработки. Этот класс также хранит экспертные знания, полученные от пользователя, т. е. список объектов класса `Seed _ Words`. Каждый из таких элементов хранит название аспектной категории (`category _ name`), образцы аспектных терминов (`words`), а также соответствующие им векторные представления (`vectors`). Основные методы этого класса решают следующие подзадачи аспектно-эмоционального анализа:

— `get _ aspect _ terms()` — выполняет извлечение аспектных терминов для каждого тестового документа;

— `detect _ polarities()` — определяет тональность каждого найденного предыдущим методом аспектного термина;

— `get _ categories _ polarity()` — для каждого тестового документа выполняет вычисление интегральных оценок тональности по каждой аспектной категории.

Класс `Extractor` решает задачу извлечения аспектных терминов в соответствии с подходом, предложенным в работе [10]. Основные методы класса:

— `extract _ aspect _ terms()` — извлечение аспектных терминов из указанного текста;

— `process _ found _ terms()` — выполнение действий постобработки для извлеченных аспектных терминов (примером такого действия является объединение терминов, написанных через предлоги, например, два извлеченных термина во фразе "*принесли **салат** с **креветками***" станут единым термином "*принесли **салат с креветками***");

— `detect _ asp _ categ4terms()` — определение аспектных категорий извлеченных терминов.

Класс `Sentimentor` отвечает за определение тональностей извлеченных аспектных терминов. Следующие методы являются ключевыми для этого класса:

— `weight _ candidates()` — осуществляет взвешивание кандидатов в эмоциональные выражения. Для вычисления весов задействуются соответствующие реализации класса взвешивания `Weighter`. Класс `Weighter _ PMI` предназначен для вычисления весов на основе метода взаимной информации PMI, класс `Weighter _ DRM` предназначен для вычисления весов на основе модели распределенных представлений слов;

— `get _ context _ vector()` — получение признакового вектора для конкретного аспектного термина;

— `predict()` — предсказание тональностей для тестовых векторов.

Результаты аспектно-эмоционального анализа представляются в виде довольно сложной лингвистической разметки. Для каждого текста необходимо хранить перечисление аспектных терминов. Каждый из таких терминов должен содержать позиционную привязку к исходному тексту, аспектную категорию, тональность термина. Помимо этого, для текста в целом необходимо хранить тональности каждой аспектной категории. Один из вариантов хранения такой разметки — XML-документы [18]. Расширяемый язык разметки (*eXtensible Markup Language,* XML) позволяет сохранять текст и сопутствующую инфор-

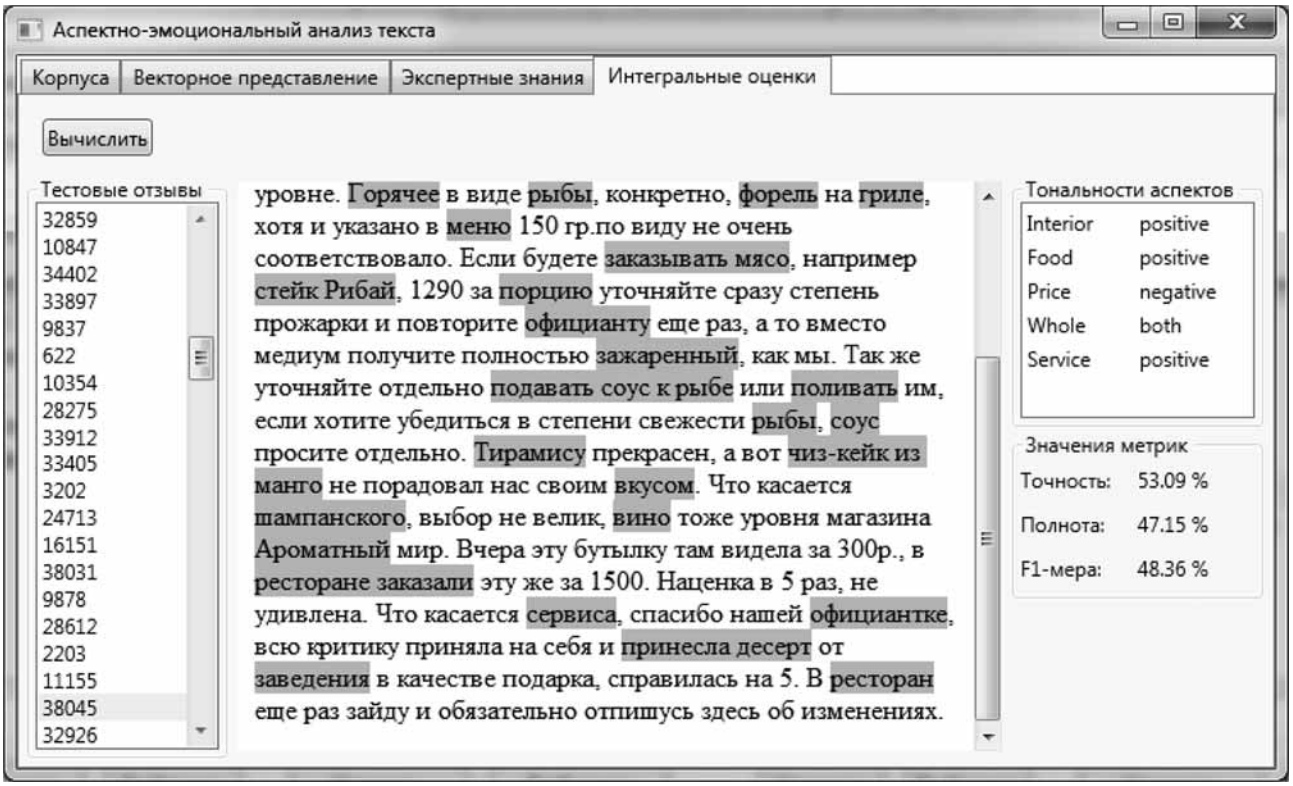


Рис. 3. Вкладка "Интегральные оценки"

мацию в одном файле. При этом аспектные термины представляются в виде отдельных XML-элементов, а вся дополнительная информация указывается в атрибутах такого элемента. Преимущество этого подхода по сравнению, например, с реляционными базами данных, в том, что не требуется специализированное программное обеспечение, например система управления базами данных. XML-файлы доступны для просмотра и редактирования даже простыми текстовыми редакторами. Для программного взаимодействия с файлами в формате XML в системе реализован класс `Xml _ Helper`.

Программная реализация обозначенных классов выполнена на платформо-независимом языке Python. Основная причина выбора этого языка — наличие инструментов обработки текстовой информации, например, библиотека NLTK [19], и библиотек машинного обучения, например Scikit-learn [20].

Интерфейсная часть выполнена для программной платформы Microsoft.NET, которая представляется одним из удобных средств при реализации графических настольных приложений для семейства операционных систем Windows. Существуют открытые реализации платформы .NET, например проект Mono [21], поэтому при необходимости интерфейсная часть системы может работать в семействе операционных систем Linux и Mac.

Взаимодействие пользователя с системой сводится к следующим четырем основным этапам:

1) выбор текстовых корпусов;

2) настройка и построение распределенных представлений слов;

3) предоставление экспертных знаний;

4) анализ и просмотр результатов.

После выполнения настройки системы (этапы 1—3) выполняется непосредственно запуск процесса анализа (этапа 4). Результаты сохраняются в виде XML-файла и становятся доступны для просмотра (рис. 3).

При этом отображается размеченный в соответствии с извлеченными терминами текст и интегральные оценки по заданным аспектным категориям. Если тестовый файл содержал эталонную разметку, дополнительно отображаются значения метрик точности, полноты и F_1-меры [15], которые удалось достичь на данном тестовом корпусе.

Экспериментальные результаты

Эффективность разработанной системы оценивалась с использованием текстовых корпусов, предоставленных в рамках российского семинара по тестированию систем анализа тональности SentiRuEval [22]. Одна из предложенных задач семинара заключалась в детальном анализе пользовательских отзывов (о ресторанах и об автомобилях), т. е. в выявлении тональностей относительно аспектов заданных объектов. Объект *ресторан* анализировался по следующим основным аспектам: *кухня*, *сервис*, *интерьер* и *цена*. Для объекта *автомобиль* интерес пред-

Результаты системы на корпусе SentiRuEval-2015 (F_1-мера, %)

Предметная область	Подзадача		
	1	2	3
Рестораны	*66,51*	*26,71*	*27,20*
	72,84 (1 из 14)	55,45 (1 из 7)	45,81 (1 из 5)
Автомобили	*69,66*	*26,48*	*23,68*
	70,16 (5 из 10)	56,84 (1 из 7)	43,90 (1 из 1)

ставляли такие аспекты, как *комфорт*, *внешний вид*, *надежность*, *безопасность*, *управляемость* и *цена*.

В таблице представлены результаты работы системы для трех подзадач, обозначенных во введении:

1) извлечение аспектных терминов;

2(определение тональности терминов;

3) определение тональности аспектных категорий.

F_1-мера является гармоническим средним двух важных показателей эффективности системы — точности и полноты [15]. Курсивом указаны значения, полученные базовыми алгоритмами (*baseline*). Для подзадачи извлечения терминов такой алгоритм заключался в переносе аспектных терминов из обучающей коллекции в тестовую. Базовые алгоритмы для подзадач определения тональности присваивали (терминам/аспектным категориям) наиболее частотную тональность, встречающуюся в обучающих коллекциях [22].

Под значениями метрик в скобках приведена позиция системы среди общего числа систем-участников для этой подзадачи и предметной области. Например, для предметной области отзывов о ресторанах в подзадаче 1 описанная система заняла 1-е место среди 14 других систем с результатом F_1-меры, равным 72,84 %.

Предлагаемая система показала лучший результат во всех подзадачах, за исключением подзадачи извлечения аспектных терминов из отзывов об автомобилях. Вероятно, сказалась особенность построения аспектных терминов этой предметной области. Например, многие термины содержали числовые обозначения и англоязычные аббревиатуры: *двигатель 2,5 литра*, *мотор 1700 DTI*, *m30b30 двигатель*, *ваз 2114*, *bmw 528i* и др. Поскольку перенос системы (из области ресторанов) осуществлялся без дополнительных изменений в блоке учета лексической структуры (см. рис. 1), поэтому специфика терминов этой предметной области не была учтена.

Таким образом, разработанная система продемонстрировала хорошие результаты при извлечении аспектных терминов и стабильно высокие результаты в задачах определения тональности.

Заключение

Предложена структура системы автоматического аспектно-эмоционального анализа текста, приведены состав подсистем и их назначение, а также элементы программной реализации.

Апробация полученной системы в рамках российского тестирования SentiRuEval показала ее эффективность в задаче извлечения мнений пользователей из отзывов, принадлежащих различным предметным областям.

Преимуществами полученной системы являются:

- слабая зависимость от обучающих данных;
- возможность извлечения аспектных терминов даже с опечатками и ошибками написания;
- точность определения тональности выражений, получающаяся путем вычисления двух взаимодополняющих оценок.

Предлагаемая система может быть использована как самостоятельный инструментальный комплекс для резюмирования мнений по аспектам исследуемой сущности на основе коллекции документов. Возможна также интеграция основной части системы в поисковые сервисы, ориентированные на подготовку рекомендаций по результатам анализа текстовых данных. Целью такой интеграции может быть учет пользовательских предпочтений относительно важности тех или иных аспектов целевых данных при формировании поисковой выдачи или выработке рекомендаций.

Работа выполнена при финансовой поддержке Министерства образования и науки РФ, государственное задание ВятГГУ (код проекта 586).

Список литературы

1. **Liu B.** Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers, 2012, 168 p.

2. **Hu M., Liu B.** Mining and Summarizing Customer Reviews// Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04). 2004. P. 168—177.

3. **Kuznetsova E. S., Loukachevitch N. V., Chetviorkin I. I.** Testing rules for a sentiment analysis system// Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". Bekasovo, 2013. Vol. 2, Issue 12, P. 71—80.

4. **König A., Brill E.** Reducing the Human Overhead in Text Categorization// Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Philadelphia, 2006. P. 598—603.

5. **RCO** Fact Extractor SDK. URL: http://www.rco.ru/?page_id=3554.

6. **Analytical** Courier. URL: http://www.i-teco.ru/solutions/business_intelligence_products/analytical_courier.

7. **Gupta D., Ekbal A.** IITP: Supervised Machine Learning for Aspect based Sentiment Analysis// Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). Dublin, Ireland, 2014. P. 319—323.

8. **Pang B., Lee L., Vaithyanathan S.** Thumbs up? Sentiment Classification using Machine Learning Techniques // Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2002. P. 79—86.

9. **AlchemyAPI,** Sentiment Analysis API. URL: http://www.alchemyapi.com/api/sentiment-analysis.

10. **Blinov P. D., Kotelnikov E. V.** Using Distributed Representation for Aspect-Based Sentiment Analysis// Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue", Bekasovo, 2014. Issue 13, P. 68—79.

11. **Блинов П. Д., Котельников Е. В.** Метод определения тональности аспектных терминов // В мире научных открытий. 2014. № 12.1 (60). С. 333—351.

12. **Mikolov T., Sutskever I., Chen K.** et al. Distributed Representations of Words and Phrases and their Compositionality // Proceedings of NIPS. 2013. P. 3111—3119.

13. **Blinov P., Kotelnikov E.** Distributed Representations of Words for Aspect-Based Sentiment Analysis at SemEval 2014// Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland. 2014. P. 140—144.

14. **Blinov P. D., Kotelnikov E. V.** Semantic Similarity for Aspect-Based Sentiment Analysis//Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". Moscow, 2015. Vol. 2, Issue 14. P. 25—34.

15. **Manning Ch., Raghavan P., Shutze H.** Introduction to Information Retrieval. Cambridge University Press, 2009. 496 p.

16. **Islam A., Inkpen D.** Second Order Co-occurrence PMI for Determining the Semantic Similarity of Words// Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006). 2006. P. 1033—1038.

17. **Унифицированный** язык моделирования UML. URL: http://www.uml.org.

18. **Extensible** Markup Language (XML). URL: http://www.w3.org/XML.

19. **Loper E., Bird S.** NLTK: The Natural Language Toolkit// Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. 2002. Vol. 1. P. 63—70.

20. **Pedregosa F., Varoquaux G., Gramfort A.** et al. Scikit-learn: Machine Learning in Python// Journal of Machine Learning Research. 2011. Vol. 12. P. 2825—2830.

21. **Project** Mono. URL: http://www.mono-project.com.

22. **Loukachevitch N. V., Blinov P. D., Kotelnikov E. V.** et al. SentiRuEval: Testing Object-oriented Sentiment Analysis Systems in Russian// Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue", Moscow. 2015. Vol. 2, Issue 14. P. 2—13.

P. D. Blinov, Software Engineer, e-mail: blinoff.pavel@gmail.com,
Vyatka State Humanities University, Kirov

Software System for Aspect-Based Sentiment Analysis

The article is devoted to the problem of automatic sentiment analysis also known as opinion mining. Aspect-based version of such analysis is given in detail; its main subtasks are listed with examples. The primary goal of the article is to show a way for opinion mining system implementation.

The first part of the article gives overall structure of a system for aspect-based sentiment analysis. The whole system can be broken into four key subsystems namely are preprocessing, subsystems of expert knowledge, linguistic resources and main control subsystem. Depending on the method being implemented the number of subsystems can vary slightly. For example, in this article there is additional word representation subsystem. It implements the method which uses semantic similarity to address key subtasks of aspect-based sentiment analysis.

The second part of the article shows how such system can be realized in practice. The system is implemented according to the object-oriented paradigm. Main classes and their relations are described with UML class diagram. Also the example of a user interface and a way of storing the results are proposed.

In the third part of the article the experimental results based on the text corpora of restaurant and car reviews, provided by seminar SentiRuEval-2015' organizers, are presented. This results show the superiority of the proposed system over similar ones.

The developed system can be used as a standalone application for document analysis. It also can be integrated within search engine or recommendation service to enrich their results with aspect-specific sentiment knowledge.

Keywords: *natural language processing, automatic sentiment analysis, aspect-based sentiment analysis, opinion mining, class diagram, software system.*

References

1. **Liu B.** *Sentiment Analysis and Opinion Mining.* Morgan & Claypool Publishers, 2012, 168 p.

2. **Hu M., Liu B.** Mining and Summarizing Customer Reviews, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, 2004, pp. 168—177.

3. **Kuznetsova E. S., Loukachevitch N. V., Chetviorkin I. I.** Testing rules for a sentiment analysis system, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, Bekasovo, 2013, vol. 2, issue 12, pp. 71—80.

4. **Konig A., Brill E.** Reducing the Human Overhead in Text Categorization, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, 2006, pp. 598—603.

5. **RCO** Fact Extractor SDK, available at: http://www.rco.ru/?page_id = 3554.

6. **Analytical** Courier, available at: http://www.i-teco.ru/solutions/business_intelligence_products/analytical_courier.

7. **Gupta D., Ekbal A.** IITP: Supervised Machine Learning for Aspect based Sentiment Analysis, *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, 2014, pp. 319—323.

8. **Pang B., Lee L., Vaithyanathan S.** Thumbs up? Sentiment Classification using Machine Learning Techniques, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002, pp. 79—86.

9. **AlchemyAPI,** Sentiment Analysis API, available at: http://www.alchemyapi.com/api/sentiment-analysis.

10. **Blinov P. D., Kotelnikov E. V.** Using Distributed Representation for Aspect-Based Sentiment Analysis, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, Bekasovo, 2014, Issue 13, pp. 68—79.

11. **Blinov P. D., Kotelnikov E. V.** Metod opredeleniya tonal'nosti aspektnykh terminov (Aspect Term Polarity Detection Method), *V Mire Nauchnykh Otkrytii*, 2014, no. 12.1 (60), pp. 333—351 (in Russian).

12. **Mikolov T., Sutskever I., Chen K., Corrado G., Dean J.** Distributed Representations of Words and Phrases and their Compositionality, *Proceedings of NIPS*, 2013, pp. 3111—3119.

13. **Blinov P., Kotelnikov E.** Distributed Representations of Words for Aspect-Based Sentiment Analysis at SemEval 2014, *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, 2014, pp. 140—144.

14. **Blinov P. D., Kotelnikov E. V.** Semantic Similarity for Aspect-Based Sentiment Analysis, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, Moscow, 2015, vol. 2, issue 14, pp. 25—34.

15. **Manning Ch., Raghavan P., Shutze H.** *Introduction to Information Retrieval.* Cambridge University Press, 2009, 496 p.

16. **Islam A., Inkpen D.** Second Order Co-occurrence PMI for Determining the Semantic Similarity of Words, *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006)*, 2006, pp. 1033—1038.

17. **Unified** Modeling Language, UML, available at: http://www.uml.org.

18. **Extensible** Markup Language (XML), available at: http://www.w3.org/XML.

19. **Loper E., Bird S.** NLTK: The Natural Language Toolkit, *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 2002, vol. 1, pp. 63—70.

20. **Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E.** Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2825—2830.

21. **Project** Mono, available at: http://www.mono-project.com.

22. **Loukachevitch N. V., Blinov P. D., Kotelnikov E. V., Rubtsova Yu. V., Ivanov V. V., Tutubalina E.** SentiRuEval: Testing Object-oriented Sentiment Analysis Systems in Russian, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, Moscow, 2015, vol. 2, issue 14, pp. 2—13.

ИНФОРМАЦИЯ

Продолжается подписка на журнал "Программная инженерия" на первое полугодие 2016 г.

Оформить подписку можно через подписные агентства или непосредственно в редакции журнала.

Подписные индексы по каталогам:

Роспечать — 22765; Пресса России — 39795

Адрес редакции: 107076, Москва, Стромынский пер., д. 4, редакция журнала "Программная инженерия"

Тел.: (499) 269-53-97. Факс: (499) 269-55-10. E-mail: prin@novtex.ru

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор З. *В.* Н*аумова*

Сдано в набор 04.09.2015 г. Подписано в печать 21.10.2015 г. Формат 60×88 1/8. Заказ PI1115
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**