МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. М. В. ЛОМОНОСОВА

**Факультет
вычислительной математики
и кибернетики**

Сборник статей молодых ученых факультета ВМиК МГУ

Выпуск 6

МОСКВА — 2009

УДК 517.6+519.8
ББК 22
С23

Составители:
А. В. ИЛЬИН, В. В. ФОМИЧЕВ, А. В. СТОЛЯРОВ

С23 **Сборник статей молодых ученых факультета ВМиК МГУ** / Ред. совет: Ложкин С. А. и др. — М.: Издательский отдел факультета ВМиК МГУ (лицензия ИД №05899 от 24.09.2001), выпуск №6, 2009 — 191 стр.

В настоящий сборник вошли статьи, выполненные молодыми учеными факультета Вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова в 2008–2009 г.г.

УДК 517.6+519.8
ББК 22

ISBN 978-5-89407-378-1

СОДЕРЖАНИЕ

Данный выпуск посвящается
девяностолетию академика
Александра Андреевича
САМАРСКОГО

УДК 681.3.06

ВОССТАНОВЛЕНИЕ ТИПОВ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИИ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРОГРАММЫ

© 2009 г. В. Ю. Антонов, А. П. Фокин, К. Н. Долгова

avadim@gmail.com, apfokin@gmail.com, katerina@ispras.ru

Кафедра системного программирования

1 Введение

Программные приложения, предоставленные сторонними разработчиками, обычно нуждаются в анализе. Даже те приложения, которые программист разрабатывает сам, в случаях критичности приложения с точки зрения информационной безопасности, требуют дополнительного анализа. Однако чем выше уровень представления программы, тем более развитые инструментальные средства разработаны для их анализа. Так, инструментальные средства для анализа бинарного кода развиты слабо, а инструментальных средств для анализа приложений на языках высокого уровня Си и Си++ достаточно много.

Направление информационных технологий, которое занимается получение неявной информации из программного кода с целью повышения уровня представления программ называется *обратной инженерией*[1]. Типичный тракт повышения уровня программного приложения начинается с бинарного кода, который дизассемблером восстанавливается в ассемблерный листинг, который декомпилятором восстанавливается в программу на языке высокого уровня, а дальше используются различные инструментальные средства, повышающие уровень представления программы до языка спецификаций.

Одним из этапов работы декомпилятора является восстановление типов данных. Качество работы декомпилятора оценивается «понятностью» кода, который он восстанавливает, то есть, чем выше качество декомпилятора, тем меньше восстановленный им код отличается от того, который бы программист написал вручную. Точность и полнота восстановления типов данных без явных операций приведения типов и использования искусственных типов объединений сильно повышает качество декомпиляции.

В декомпиляторе TyDec, который разрабатывается в Институте системного программирования совместно с кафедрой системного программирования ВМК МГУ имеется модуль, который восстанавливает типы данных, основываясь только на статической информации, то есть только по ассемблерному листингу.

Существуют случаи, для которых статический анализ не позволяет восстановить типы данных однозначно. Так, в примере, представленном на рис. 1, при замене типа `int` переменной s на тип `char *` ассемблерный код не изменяется. Следовательно, декомпилятор не сможет однозначно восстановить тип переменной s при статическом анализе.

```
int f(int *a, int n) {
int s = 0;
int *p = a;
for (; p < a + n; ++p) {
  s += *p;
}
return s;
}
```

Рис. 1: Си программа, для которой неоднозначно восстановление типа переменной s.

Один из возможных вариантов разрешения такого рода неоднозначностей — спросить пользователя, другой — использовать информацию, собранную в процессе работы программы на некотором наборе тестовых данных в процессе восстановления. Собранную информацию назовем профилем программы, а процесс сбора — профилированием.

Профиль программы можно использовать как дополнение к статическим алгоритмам, когда статической информации просто недостаточно.

Данная работа имеет следующую структуру. В разделе 2 предоставляется обзор работ схожей тематики. Раздел 3 посвящен описанию предлагаемого метода сбора и анализа информации времени выполнения программы. В разделе 5 представлено описание системы `Valgrind` [2], используемой для реализации предлагаемого метода. В раздел 6 представлена реализация предлагаемого метода, а в разделе 7 представлены результаты апробации представленного метода. В заключении сформулированы выводы работы и направления дальнейших исследований.

2 Обзор работ схожей тематики

Существует большое количество методов и инструментальных средств, в которых они реализованы, которые позволяют извлечь неявную информацию о типах данных для анализа программы или повышения уровня ее представления. На практике используют методы, извлекающие неявную информацию о типах данных, как в статическом режиме, так и во времени выполнения.

В работе [4] представлен метод динамического восстановления абстрактных типов данных. Цель восстановления абстрактных типов состоит в том, чтобы сгруппировать переменные не только по размеру в памяти и типу операций, которыми они обрабатываются в ассемблерном коде, но и по особенностям их использования в программе. Наличие таких «укрупненных» типов данных позволяет лучше понимать назначение переменных в программе, что существенно упрощает ее поддержку и внесение изменений. Описываемый метод основан на том, что изначально каждому значению во время выполнения программы ставится в соответствие один абстрактный тип. Во время выполнения программы собирается информация о значениях, которые принимали все переменные, после чего в результате пересечения значений переменных они разделяются на укрупненные абстрактные типы данных. Информацию о времени выполнения программы собирается посредством утилиты, которая реализована с помощью системы `Valgrind`.

В работе [3] представлено инструментальное средство *hobbes*. Этот инструмент разработан для идентификации ошибок в программах на языке Си, которые могут возникнуть в результате неправильного использования особенностей языка. Утилита интерпретирует бинарный код, используя информацию о значениях, расположенных по адресам памяти, которые встречаются в регистрах и переменных. В качестве результата утилита выдает предупреждения, когда вероятность ошибки во время выполнения программы превышает установленную пользователем границу. Для анализа используется только динамический анализ, даже для тех случаев, когда статического анализа может быть достаточно. Однако в инструментальном средстве *hobbes* слабо развита обработка указателей, а именно, утилита не различает указатели на разные типы данных, вследствие чего ошибки времени выполнения, которые могут возникнуть из-за неправильной работы с указателями, не выявляются. Сбор информации о значениях, которые принимают переменные во время выполнения, выполняется системой динамически. Инструментальное средство состоит из двух частей: интерпретатора кода `x86` и модуля проверки типов. Интерпретатор кода работает как инструментирующий компилятор, который встраивает в код вызовы функций модуля проверки типов. Архитектура интерпретатора аналогична инструментирующему компилятору, на основе которого построена система `Valgrind`. Само инструментальное средство `Valgrind` в реализации не использовано, так как на момент начала разработки еще не существовало его работающей версии.

3 Профилирование значений ячеек памяти

В декомпиляторе *TyDec* реализованы мощные методы статического восстановления типов данных. В случаях, когда статических методов недостаточно для повышения точности восстановления типов данных можно использовать информацию времени выполнения программ, а именно, множество значений, которые хранились в ячейках памяти и регистрах процессора во время выполнения программы. В частности, можно с достаточной степенью достоверности определить, что некоторый регистр в некоторой точке программы хранит 32-х битное значения, которые являются адресами, что соответствует переменной указательного типа в исходной программе на языке Си, даже если она не разыменовывалась. Подобным образом можно сделать обоснованное предположение о знаковости значений в регистре и, соответственно, о знаковости типа соотвествующей этому регистру переменной в исходной программе на языке Си.

Схема виртуального адресного пространства процесса для ОС Linux на 32-х битной архитектуре IA32 представлена на рис. 2. Каждому процессу выделяется диапазон виртуальных адресов от `0x00000000` до `0xFFFFFFFF` Операционная система выделяет процессу 3Гб, которые расположены непрерывно, начиная с ячейки с адресом `0x00000000` и до ячейки с адресом по `0xBFFFFFFF`. Оставшаяся память зарезервирована системой. Начиная с адреса `0x08048000` загружается копия исполняемого файла, сегмент данных и BSS сегмент. Кроме этого, в памяти расположены стек, куча и библиотеки. Следовательно, переменная указательного типа во время работы программы может принимать не любые значения, а только те, которые соответствуют адресам ячеек памяти виртуального адресного пространства процесса. Обозначим $ValidAddr$ — множество всех используемых адресов памяти программы. Очевидно, что оно включает в себя все перечисленные сегменты, используемые процессом.

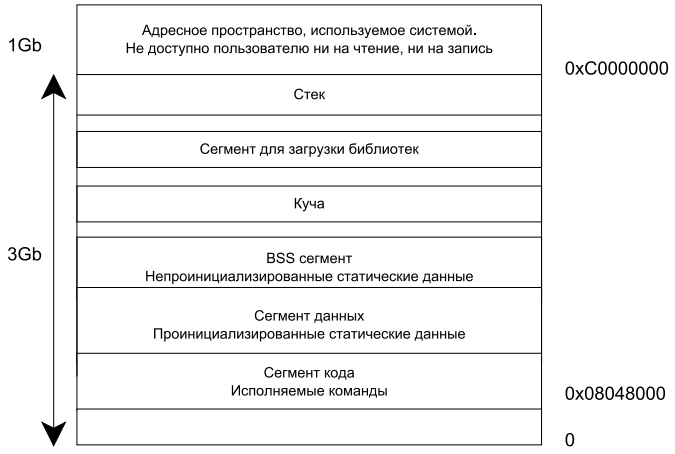


Рис. 2: Виртуальное адресное пространство процесса.

Обозначим через *obj* конструкции ассемблерной программы, соотвествующие переменным на языке высокого уровня. То есть это:

- регистры общего назначения центрального процессора,
- ячейки памяти в абсолютной адресации, которые соответствуют глобальным переменным в исходной программе,
- ячейки памяти по фиксированным смещениям относительно стекового кадра, соответствующие локальным параметрам,
- ячейки памяти по фиксированному смещению относительно стекового кадра по регистру %esp, а также положенные на стек соответствующими инструкциями. Они соответствуют фактическим параметрам в вызываемых функциях.

В корректной программе на языке Си указатели, как правило, хранят адреса из множества отображенных виртуальных адресов (стек, куча, статические данные, код), либо специальное значение *null*. Значения, которые принимают знаковые целочисленные переменные, вне зависимости положительные они или отрицательные, сгруппированы около нуля, то есть значение -1 переменная принимает значительно чаще, нежели, например, $-2 \cdot 10^9$. Беззнаковые целочисленные переменные редко принимают значения, которые соответствуют -2 или -3 а, например, значение $3 \cdot 10^9$ переменные принимают чаще для 32-х битной архитектуры, нежели беззнаковое значение, соответствующее -2. Значение -1 необходимо рассматривать специальным образом, так как оно соответствует максимальному беззнаковому целому значению и часто выступает в роли индикатора ошибки.

Для 32-х битной архитектуры нужно профилировать значения только для 32-х битных ячеек памяти, так как указатели имеют размер 32-бита. Несмотря на то, что в идеале хорошо бы профилировать значения напрямую ячеек памяти, это на практике реализовать затруднительно, так как:

- значения разных переменных могут храниться в одних по одним и тем же адресам. Стек, как и куча, постоянно переиспользуются,
- одни и те же локальные переменные могут иметь различный адрес в разные моменты времени.

Однако для того чтобы использовать значение ячейки памяти ее необходимо загрузить на регистр. Следовательно, вместо профилирования непосредственно значений ячеек памяти, будем собирать значения, которые загружаются на регистр и выгружаются из регистров соответствующими командами.

Пусть VP — это отображение из множества $Addr$ виртуальных адресов сегмента кода программы во множество 32-х битных значений Val, которые записывались или считывались инструкцией по этому адресу. Множество адресов, которые не принадлежат сегменту кода программы, исключены из рассмотрения.

Виртуальные адреса должны быть отображены на инструкции дизассемблированной программы. Следовательно, отображение DM — это отображение из множества $Addr$ виртуальных адресов памяти в множество объектов Loc, соответствующих этим адресам памяти в дизассемблированной программе.

Отображение IVP из множества ассемблерных инструкций во множество 32-х битных значений, которые загружаются на регистр или выгружаются в память соответствующими инструкциями, строится посредством комбинирования отображений VP и DM

$$VP : Addr \to 2^{Val}$$
$$DM : Addr \to Loc$$
$$IVP : Loc \to 2^{Val}$$
$$IVP(x) = VP(DM^{-1}(x))$$

В качестве результата получаем, что каждой инструкции дизассемблированной программы ставится в соответствие множество 32-х битных значений, которое она загружает или выгружает в память. Если значение, которое загружается или выгружается не 32-х битное, или оно не было покрыто во время выполнения программы, оно считается не аннотированным.

Множество $VP(obj)$ — это множество значений, принимаемых объектом obj за время выполнения программы. Другими словами, это — множество значений, записываемых в ячейку памяти, выделенную для переменной, соответствующей объекту obj.

$null$ — это нулевой указатель.

$\sigma_{obj}(x)$ — функция, возвращающая 1, если $x \in VP(obj)$, и 0 в противном случае.

Рассмотрим две метрики:

$PC(obj)$ — мера уверенности «указатель». Принимает значения от 0 до 1. Чем ближе значение к 1, тем больше уверенность в том, что объект obj является указателем.

$$PC(obj) = \frac{|VP(obj) \cap ValidAddr|}{|VP(obj) - null|}$$

Так как указатель может принимать значение *null*, которое не включено в $ValidAddr$, *null* исключается из множества значений, принимаемых объектом *obj*.

$UC(obj)$ — мера уверенности «беззнаковый» для 32-х битной платформы. Принимает значения от $\frac{1}{2^{31}}$ до 1. Чем ближе значение меры к 0, тем больше уверенности, что переменная *obj* является знаковой.

$$UC(obj) = 1 - \sum_{x=-2^{31}+1}^{-2} \frac{\sigma_{obj}(x)}{2^{2*\lfloor log_2|x+1|\rfloor+1}}$$

Если переменная принимала значение -2, то мера уверенности «беззнаковый» уменьшается на $1/2$. Для значений -3 и -4 мера уверенности «беззнаковый» уменьшается на $1/4$. Для остальных отрицательных значений производятся аналогичные действия. Из этого следует метрика и её минимальное значение:

$$1 - \sum_{x=-2^{31}+1}^{-2} \frac{1}{2^{2*\lfloor log_2|x+1|\rfloor+1}} = \frac{1}{2^{31}}$$

Данные метрики положены в основу метода сбора и анализа информации времени выполнения программы для восстановления знаковости целочисленной переменной и отличия целого типа данных от указательного по бинарному коду исполняемой программы. Метод основан на построении профиля значений для каждой ячейки памяти в программе, соответствующей переменной, и вычисления для них метрик. На основе полученных значений, можно предположить, является ли переменная знаковой или беззнаковой, целочисленного типа или указательного.

4 Распознование инструкций memset и memcpy

Стандарт языка Си позволяет выполнять побайтовый доступ к объектам. Так, например, возможно скопировать значение одного структурного типа в другой структурный тип побайтно, используя функцию `memcpy`. Обычно компилятор вставляет целиком (*inline*) функции побайтового копирования `memcpy`, побайтового обнуления `memset` и тому подобные, вместо их вызова для оптимизации. Тела таких функций должны быть идентифицированы в декомпилируемом коде и не рассматриваться при восстановлении типов, так как они нарушают правила строгой типизации и правила распротстанения информации о типах по программе.

Распознавание инструкций `memset` и `memcpy` можно проводить либо сигнатурным анализом, либо анализом поведения во время выполнения программы. Сигнатурный анализ требует постоянно обновляющуюся базу сигнатур для всех поддерживаемых компиляторов. Анализ поведения во время выполнения программы проще при реализации и использовании.

Инструкции `memset` и `memcpy` в результате оптимизации компиляторов могут быть заменены набором идущих подряд считываний и присваиваний (см. Таблицу 2).

Например, функция `memset(p, 0, 2 * sizeof(*p))` транслируется в ассемблер в виде кода, состоящего из двух инструкций:

```
movl   $0 , (%eax )
movl   $0 , 4(%eax )
```

Функция `memcpy(q, p, 5 * sizeof(*p))` транслируется как

```
movl  (%edx), %eax
movl  %eax, (%ecx)
movl  4(%edx), %eax
movl  %eax, 4(%ecx)
movl  8(%edx), %eax
movl  %eax, 8(%ecx)
movl  12(%edx), %eax
movl  %eax, 12(%ecx)
movl  16(%edx), %eax
movl  %eax, 16(%ecx)
```

Таким образом, если во время анализа выполнения программы были найдены идущие подряд инструкции присваивания `movl` $x, addr_i$, для которых:

- Используется фиксированное значение x.
- Адреса ячеек памяти $addr_i$ расположены в возрастающем порядке или в убывающем и $|addr_i - addr_{i-1}|$ =размер ячейки памяти.

Тогда этот набор инструкций можно заменить на инструкцию `memset`, которая присваивает значение x ячейкам памяти, расположенным по адресам $addr_i$.

Если во время анализа выполнения программы были найдены чередующиеся инструкции считывания `movl` $addr1_i, eax$ и присваивания `movl` $eax, addr2_i$, для которых:

- Адреса ячеек памяти $addr1_i$ расположены в возрастающем порядке или в убывающем и $|addr1_i - addr1_{i-1}|$ =размер ячейки памяти.
- Адреса ячеек памяти $addr2_i$ расположены в порядке, соответствующем порядку ячеек памяти $addr1_i$, и $|addr2_i - addr2_{i-1}|$ =размер ячейки памяти.

Тогда этот набор инструкций можно заменить на инструкцию `memcpy`, которая копирует значения из ячеек памяти $addr1_i$ в ячейки памяти $addr2_i$.

5 Система Valgrind

При разработке утилит динамического анализа исполняемых файлов можно использовать для упрощения разработки системы построения инструментальных систем для динамического анализа. Особенностью таких систем является то, что для интеграции их с анализируемым приложением не требуется исходного кода приложения, а достаточно только исполняемого модуля. Одной из таких систем является система построения динамических инструментов для исполняемых файлов `Valgrind`

Система `Valgrind` позволяет писать утилиты на основе своего ядра, которые позволяют анализировать бинарные приложения. Особенностью системы являются развитая функциональность устойчивость приложений к ошибкам времени выполнения, высокая производительность за счет минимизации количества операций.

Инструментальное средство представляет собой надстройку, реализованную на языке Си, к ядру системы `Valgrind` В общем виде инструментальное средство можно представить следующим образом: *Valgrind core + tool plug-in = Valgrind tool*. Инструментальное средство статически линкуется к исполняемому модулю, в который добавляется дополнительный код для вызова методов ядра системы `Valgrind`.

На основе системы `Valgrind` реализовано много инструментальных систем, позволяющих выполнять динамический анализ приложений.

1. Система `Memcheck`[5] позволяет проводить анализ всех операций чтения и записи в памяти, выделение и освобождение блоков памяти. В результате, система позволяет выявлять следующие ошибки в приложении:
   (a) использование непроинициализированной памяти,

(b) чтение/запись памяти после её освобождения,

(c) чтение/запись несоответствующих адресов в стеке,

(d) утечки памяти,

(e) неправильное использование операторов `malloc`/`new`/`new []` и `free`/`delete`/`delete []`,

(f) перекрывающиеся области памяти в `src` и `dst` в `memcpy()` и других аналогичных функциях.

2. Система `Cachegrind`[2] позволяет находить места в программе, где имеется не оптимальное взаимодействие с типичным суперскалярными процессорами, в следствие чего время работы программы сильно увеличивается. Система симулирует программу и добавляет комментарии в исходный код об отсутствии нужных данных в кэше или неправильном прогнозировании ветвления.

3. Система `Callgrind`[2] позволяет построить граф вызовов для программного запуска. По умолчанию, система собирает информацию о количестве выполнившихся инструкций, количество вызовов между функциями, а так же считает отношение долю выполнившихся инструкций по отношению к инструкциям программы. Помимо этого симулятор кэша представляет информацию о поведении операций доступа к памяти.

4. Система `Helgrind` [2] позволяет выполнять анализ с целью выявления ошибок синхронизации потоков в программах, реализованных на языках Си, Си++ и Fortran, использующих примитивы стандарта POSIX.

Архитектура системы `Valgrind` основана на методе «Дизассемблирование и повторный синтез», который позволяет встраивать код анализа в исполняемый код программы. Реализация этого метода системе состоит из 8 этапов:

1. Дизассемблирование. На этом этапе исполняемый код программы преобразуется во внутреннее представление системы `Valgrind` в виде древовидной структуры.

2. Оптимизация, этап 1. Внутреннее представление оптимизируется и преобразуется в список.

3. Инструментация. На этом этапе выполняется добавление инструментированного кода во внутреннее представление программы.

4. Оптимизация, этап 2. На этом этапе выполняется дополнительное оптимизационное преобразование внутреннего представления программы.

5. Построение дерева. После того как выполнены оптимизационные преобразования внутреннее представление программы из списка перестраивается обратно в древовидную структуру. В течение этого шага ВП преобразуется в древовидную структуру.

6. Выделение инструкций. На этом этапе по внутреннему представлению в виде древовидной структуры выделяются инструкции, которые преобразуется в список.

7. Выделение регистров. Из списка инструкций перестраивается политика использования регистров, что необходимо в результате добавления инструментированного кода.

8. Сборка. На последнем этапе список инструкций преобразуется в машинный код.

Так как система `Valgrind` для инструментации не требует исходного кода и является продуктом, который распространяется свободно, она была выбрана для реализации утилиты, позволяющей собирать и анализировать информацию о типах данных, доступную только во время выполнения приложения.

Рис. 3: Общая схема работы модуля восстановления типов данных декомпилятора TyDec.

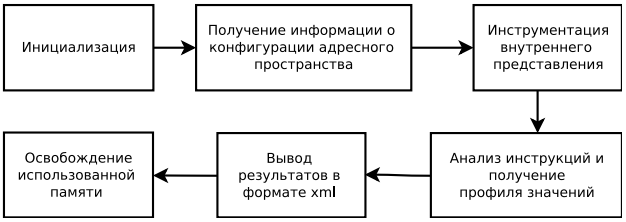


Рис. 4: Схема работы инструмента TDTrace

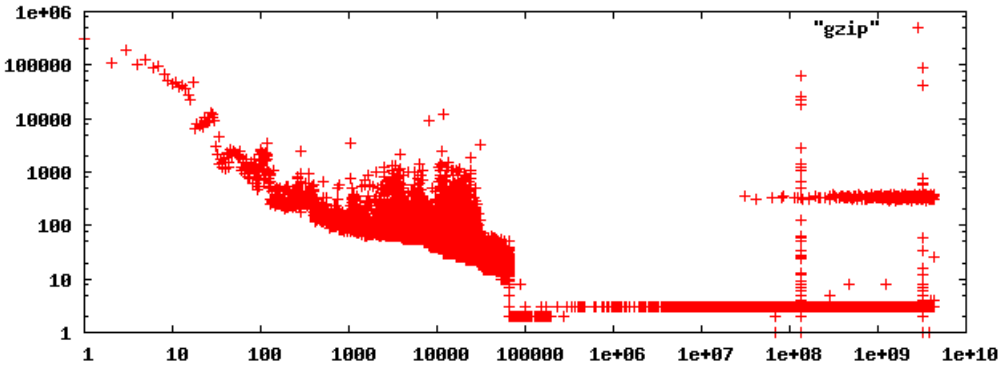


Рис. 5: Набор полученных значений ячеек памяти для программы gzip

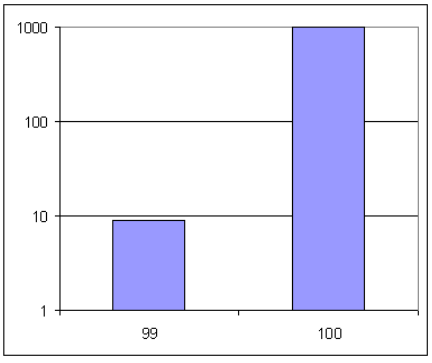


Рис. 6: Статистика полученных значений метрики $UC(obj)$ для программы gzip

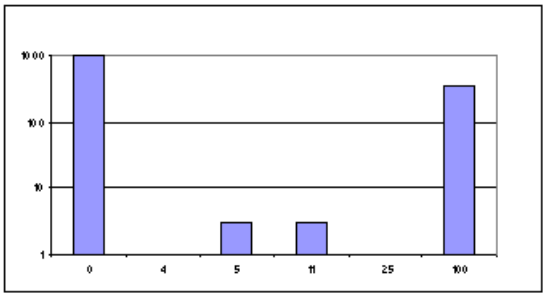


Рис. 7: Статистика полученных значений метрики $PC(obj)$ для программы gzip

```
...
<memoryAddress type="variable" addr="0x0804A008" pointer="0" />
<memoryAddress type="variable" addr="0x0804A00C" pointer="0" />
<memoryAddress type="variable" addr="0x0804A010" pointer="0" />
<memoryAddress type="variable" addr="0x0804A014" pointer="0" />
<memoryAddress type="variable" addr="0x0804A018" pointer="0" />
<memoryAddress type="variable" addr="0x0804A01C" pointer="0" />
<memoryAddress type="variable" addr="0x0804A020" pointer="0" />
<memoryAddress type="variable" addr="0x0804A024" pointer="0" />
<memoryAddress type="variable" addr="0x0804A028" pointer="0" />
<memoryAddress type="variable" addr="0x0804A02C" pointer="0" />
<memoryAddress type="variable" addr="0xBE9754A0" pointer="100" />
<memoryAddress type="variable" addr="0xBE9754A4" pointer="0" />
<memoryAddress type="variable" addr="0xBE9754A8" pointer="100" />
<memoryAddress type="variable" addr="0xBE9754B0" pointer="100" />
<memoryAddress type="variable" addr="0xBE9754B4" pointer="0" />
<memoryAddress type="variable" addr="0xBE9754D0" pointer="100" />
...
```

Таблица 1: Пример вывода утилиты TDTrace

6 Утилита TDTrace

На рис. 3 представлена общая схема работы модуля, восстанавливающего типы данных декомпилятора *TyDec*.

На вход модулю, выполняющему восстановление типов данных, подается ассемблерная программа в виде внутреннего представления. По внутреннему представлению строится начальное множество объектов, которое подается на вход алгоритму восстановления базовых типов данных. Результатом работы алгоритма является три группы объектов:

1. объекты, для которых базовые типы восстановлены,
2. объекты, для которых восстановлена информация о том, что они имеют некоторый указательный тип,
3. Объекты, для которых на данной итерации восстановить тип не удалось.

Объекты указательного типа передаются на вход алгоритму восстановления производных типов данных. Результатом работы этого алгоритма является новое множество объектов, построенное для каждого поля скелета производного типа и для первого элемента массива. Это новое множество объектов объединяется с множеством объектов, типы которых не были восстановлены алгоритмом восстановления базовых типов данных и передается опять на вход алгоритму восстановления базовых типов. Если у структуры все типы полей восстановлены, то такая структура считается восстановленной иотправляется во множество восстановленных типов. Массив считается восстановленным, если восстановлен тип первого элемента и размер массива.

Информация, собранная на основе профильного анализа, хранится отдельно и подгружается по запросу. Для профилирования разработана утилита `TDTrace`

Утилита `TDTrace` позволяет анализировать адреса и значения, используемые во время выполнения программы, а так же представляет полученную информацию в удобном для последующего использования виде. Утилита получает информацию о конфигурации адресного пространства, через ядро системы `Vagrind` анализирует все инструкции программы, составляя для каждого адреса набор записываемых туда значений, считает для каждого адреса характеристики $UC(obj)$ и $PC(obj)$ и выдаёт полученный результат в формате xml-документа.

Схема работы утилиты представлена на рис. 4.

Для удобства работы с утилитой были введены параметры запуска и реализован модуль, преобразующий полученный после работы инструмента xml-документ в объектную модель.

Параметры командной строки утилиты приведены ниже:

- Опции командной строки запуска. Каждая опция может принимать значение `yes` или `no`. Значение `yes` соответствует работе функции, значение `no` означает отключение функции.
  - Опция `print-all-readwrite-instructions` отвечает за вывод адресов всех инструкций считывания и записи в память.
  - Опция `print-all-ips` отвечает за вывод адресов всех инструкций программы и списка значений, принимаемых параметрами, для каждой из них.

7 Апробация

Утилита `TDTrace` была апробирована на нескольких программах с открытыми исходными кодами.

Ниже на рис. 5, рис. 6, рис. 7 приведены результаты работы утилиты `TDTrace` для программы gzip. На рис. 5 по оси x расположены значения, принимаемые ячейками памяти, по оси y расположена величина, отражающая частоту, с которой ячейки памяти принимали это значение. На рис. 6 и рис. 7 по оси x расположены значения соответствующих метрик, а по оси y расположены относительные частоты.

Проанализировав код программы gzip было установлено, что утилита `TDTrace` верно определила все переменные указательного и беззнакового типов.

Эксперименты показали хорошие результаты, что позволяет говорить о состоятельности подхода.

8 Заключение

В данной работе рассмотрены подходы к восстановлению типов данных в задаче декомпиляции и предложен метод сбора и анализа информации времени выполнения программы для восстановления типов данных по бинарному коду исполняемой программы. Дано описание системы `Valgrind`, используемой для реализации предлагаемого метода. В работе представлено описание инструментального средства `TDTrace`, реализующего предлагаемый метод, и результаты его тестирования. Результаты тестирования показали состоятельность предложенных метрик.

Представляется актуальным интеграция разработанного инструмента в инструментальную среду декомпиляции программ. Это позволит улучшить процесс восстановления типов данных. Улучшение полученных методик и интеграция в инструментальную среду декомпиляции программ является направлением дальнейших исследований авторов.

Список литературы

[1] Chikofsky, E.J.; J.H. Cross II (January 1990) *Reverse Engineering and Design Recovery: A Taxonomy in IEEE Software.* IEEE Computer Society

[2] Nicholas Nethercote and Julian Seward. (2007) *Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation.* Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007), San Diego, California, USA, June 2007.

[3] M. Burroes, S. Fruend, J. Wiener (2003) *Run-time type checking for binary programs* Proceedings of CC, p.90-105.

[4] P. Cuo, J. Perkins,S. McCamant and M. Ernst (2006) *Dynamic inference of abstruct types.* Proceedings of ISSTA, p.749-754.

[5] Описание системы *Memcheck*: http://hald.dnsalias.net/projects/memcheck/

[6] Описание систем, реализованных на основе системы Valgrind: http://valgrind.org/info/tools.html

УДК 517.955.8

АСИМПТОТИКА ПРИ БОЛЬШИХ ВРЕМЕНАХ РЕШЕНИЯ ЗАДАЧИ КОШИ ДЛЯ УРАВНЕНИЯ СОБОЛЕВСКОГО ТИПА С АНАЛИТИЧЕСКОЙ НЕЛИНЕЙНОСТЬЮ

© 2009 г. А. И. Аристов

ai_aristov@mail.ru

Работа посвящена изучению асимптотики при больших временах обобщенного решения следующей задачи Коши:

$$\begin{cases} \frac{\partial}{\partial t}\left(\Delta u - u\right) + a\left(\Delta u - u\right) + F(u) = 0 \\ u\left(x, 0\right) = u_0\left(x\right). \end{cases} \tag{1}$$

Здесь u– действительнозначная функция, зависящая от действительного N-мерного ($N \geqslant 1$) вектора пространственных переменных x и времени $t > 0$.

$$F(u) = \sum_{n=k}^{\infty} \mu_n u^n,$$

где k – натуральное число, не равное единице. Коэффициенты $a,\ \mu_n \in R$ не зависят от координат и времени, при этом $a > 0$. Будем предполагать, что существует такое $\rho_0 > 0$, что ряд $\sum_{n=k}^{\infty} |\mu_n| \cdot |\rho|^n$ сходится при $|\rho| < \rho_0$.

Будет показано, что если начальные данные достаточно малы в смысле некоторой нормы, то обобщенное решение существует и единственно, причем нелинейность не влияет на качественное поведение решения, определяя только порядок стремления к нулю остаточного члена.

Замена $u = ve^{-at}$ сводит задачу к следующей:

$$\begin{cases} \frac{\partial}{\partial t}\left(\Delta v - v\right) + \sum\limits_{n=k}^{\infty} \mu_n e^{-(n-1)at} v^n = 0 \\ v\left(x, 0\right) = u_0\left(x\right). \end{cases} \tag{2}$$

Введем обозначения, необходимые для изложения рассуждений:

$L_1 = L_1\left(R^N\right),\ L_2 = L_2\left(R^N\right),\ L_\infty = L_\infty\left(R^N\right), H^2 = H^2\left(R^N\right),$
$\overline{\mathrm{B}}\left[\varphi\right] = F^{-1}\left[\frac{1}{1+|p|^2} F\left[\varphi\right]\right],$
$B\left(x\right) = \frac{1}{(2\pi)^N} \int\limits_{R^N} \frac{e^{i(p,x)}}{1+|p|^2} dp,$
$b = \|B\left(\cdot\right)\|_{L_1}$ *(ниже будет доказано, что такое число существует)*,
$U = L_\infty \bigcap H^2,$
$X = C\left[0;\ \infty;\ U\right),$
$\|\varphi\| = \sup\limits_{t \geqslant 0} \left(\|\varphi\|_{L_\infty} + \|\varphi\|_{H^2}\right)$
(в частности, если φ *не зависит от* t*, то* $\|\varphi\| = \|\varphi\|_{L_\infty} + \|\varphi\|_{H^2}$*)*,
$X_\rho = \{\varphi \in X \mid \|\varphi\| < \rho\}$

Преобразование Фурье вводится по правилу

$$\hat{f}\left(p\right) = F\left[f\left(x\right)\right] = \frac{1}{(2\pi)^{N/2}} \int\limits_{R^N} e^{-i(p,x)} f\left(x\right) dx.$$

Идентификатор "c", помеченный целым нижним индексом, обозначает некоторую положительную константу.

Будем предполагать, что $u_0(x) \in U$.

Лемма 1. *Существуют такие постоянные* $c_{1,2} > 0$, *что*

$$1.\ |B(x)| \leqslant c_1 |x|^{(1-N)/2} e^{-|x|},\ |x| \geqslant 1,$$
$$2.\ |B(x)| \leqslant c_2 \int\limits_{|x|}^{1} y^{1-N} dy,\ |x| < 1.$$

Лемма 2. *Существуют такая постоянная* $c_3 > 0$, *что*

$$\left\| \overline{\mathrm{B}}[\varphi] \right\|_{H^2} \leqslant c_3 \|\varphi\|_{L_2}$$

Леммы 1 и 2 доказаны в [1].

Лемма 3. *Существует конечное число* $b \equiv \|B(\cdot)\|_{L_1}$.

По определению нормы в L_1

$$\|B(\cdot)\|_{L_1} = \int\limits_{|x|\geqslant 1} |B(x)|\, dx + \int\limits_{|x|\leqslant 1} |B(x)|\, dx$$

Воспользуемся леммой 1:

$$b \leqslant c_1 \int\limits_{|x|\geqslant 1} |x|^{(1-N)/2} e^{-|x|} dx + c_2 \int\limits_{|x|\leqslant 1} \int\limits_{|x|}^{1} y^{1-N} dy dx$$

С помощью перехода к полярным координатам легко убедиться, что оба интеграла сходятся. Следовательно, $b < \infty$. **Лемма доказана.**

Лемма 4. $\overline{\mathrm{B}}[\varphi] = \int_{R^N} B(x-y)\varphi(y)\, dy$.

Утверждение следует непосредственно из определения $\overline{\mathrm{B}}[\cdot]$.

Лемма 5. *Пусть* $\varphi \in X$. *Если* $n \geqslant k \geqslant 2$, *то* $\sup\limits_t \left\| \int\limits_0^t |\varphi(x,\tau)|^n e^{-(n-1)a\tau} d\tau \right\|_{L_2} \leqslant \frac{1}{(k-1)a} \sup\limits_t \|\varphi\|_{L_\infty}^{n-1} \sup\limits_t \|\varphi\|_{H^2}$.

Действительно,

$$\sup_t \left\| \int_0^t |\varphi(x,\tau)|^n e^{-(n-1)a\tau} d\tau \right\|_{L_2} =$$

$$= \sup_t \sqrt{\int_{R^N} \left(\int_0^t |\varphi(x,\tau)|^n e^{-(n-1)a\tau} d\tau \right)^2 dx} \leqslant$$

$$\leqslant \sup_t \sqrt{\int_{R^N} \sup_t \|\varphi\|_{L_\infty}^n \int_0^\infty e^{-(n-1)a\tau} d\tau \int_0^t |\varphi|^n e^{-(n-1)a\tau} d\tau dx} \leqslant$$

$$\leqslant \frac{1}{\sqrt{(n-1)a}} \sup_t \|\varphi\|_{L_\infty}^{n/2} \sup_t \sqrt{\int_{R^N} \int_0^t |\varphi(x,\tau)|^{n-2} |\varphi(x,\tau)|^2 e^{-(n-1)a\tau} d\tau dx} \leqslant$$

$$\leqslant \frac{1}{\sqrt{(n-1)a}} \sup_t \|\varphi\|_{L_\infty}^{n-1} \sup_t \sqrt{\int_{R^N} \int_0^t |\varphi(x,\tau)|^2 e^{-(n-1)a\tau} d\tau dx} =$$

$$= \frac{1}{\sqrt{(n-1)a}} \sup_t \|\varphi\|_{L_\infty}^{n-1} \sup_t \sqrt{\int_0^t e^{-(n-1)a\tau} \|\varphi\|_{L_2}^2 d\tau} \leqslant$$

$$\leqslant \frac{1}{\sqrt{(n-1)a}} \sup_t \|\varphi\|_{L_\infty}^{n-1} \sup_t \|\varphi\|_{L_2}^2 \sqrt{\int_0^\infty e^{-(n-1)a\tau} d\tau} =$$

$$= \frac{1}{(n-1)a} \sup_t \|\varphi\|_{L_\infty}^{n-1} \sup_t \|\varphi\|_{L_2} \leqslant$$

$$\leqslant \frac{1}{(n-1)a} \sup_t \|\varphi\|_{L_\infty}^{n-1} \sup_t \|\varphi\|_{H^2} \leqslant \frac{1}{(k-1)a} \sup_t \|\varphi\|_{L_\infty}^{n-1} \sup_t \|\varphi\|_{H^2}$$

Лемма доказана.

Формально применим к уравнению в задаче (2) оператор $\overline{\mathrm{B}}[\cdot]$ и проинтегрируем от 0 до t с учетом начальных условий. Получим следующее интегральное уравнение:

$$v(x,t) = u_0(x) + \int_0^t \sum_{n=k}^\infty \mu_n e^{-(n-1)a\tau} \overline{\mathrm{B}}[v^n] d\tau \tag{3}$$

Из дальнейших рассуждений будет видно, что в этом выражении можно менять местами интегрирование и суммирование.

Определение 1. *Обобщенным решением задачи (2) называется решение уравнения (3) из пространства X.*

Определение 2. *Обобщенным решением задачи (1) называется элемент u пространства X, определяемый по правилу $u = ve^{-at}$, где v – обобщенное решение задачи (2). (Заметим, что если $v \in X$, то и $u \in X$.)*

Теорема 1. *Существует такая окрестность нуля, что если $\|u_0\|$ находится в этой окрестности, то существует единственное обобщенное решение $v \in X$ задачи (2-3) (а значит, и решение u задачи (1) из того же класса).*

Таким образом, если начальные данные достаточно малы в смысле нормы пространства X, то исследуемая задача однозначно разрешима.

Для доказательства положим $\rho = \frac{1}{2}\|u_0\|$. Обозначим

$$Mv = u_0(x) + \sum_{n=k}^{\infty} \mu_n \int_0^t e^{-(n-1)a\tau}\, \overline{\mathrm{B}}\left[v^k\right] d\tau$$

Докажем, что оператор M переводит шар X_ρ в себя и является сжимающим отображением. Тогда по принципу сжимающих отображений можно будет утверждать, что $\forall u_0 \in X_\rho$ существует единственное решение уравнения $Mw = w$ из пространства X.

Сначала докажем, что если $v \in X_\rho$, то и $Mv \in X_\rho$.

Заметим, что

$$\sup_t \|Mv\|_{L_\infty} \leqslant \|u_0\|_{L_\infty} + \sum_{n=k}^{\infty} |\mu_n| \sup_t \left\| \int_0^t e^{-(n-1)a\tau}\, \overline{\mathrm{B}}\left[v^k\right] d\tau \right\|_{L_\infty}$$

Кроме того,

$$\left\|\overline{\mathrm{B}}\left[v^n\right]\right\|_{L_\infty} = \left\| \int_{R^N} B(x-y)\, v^n(y)\, dy \right\|_{L_\infty} \leqslant$$
$$\leqslant b\|v^n\|_{L_\infty} = b\|v\|^n_{L_\infty} \leqslant b\rho^n$$

Здесь было использовано соотношение $\left\|\overline{\mathrm{B}}\left[v^n\right]\right\|_{L_\infty} \leqslant b\|v^n\|$, вытекающее из неравенства Юнга. Поэтому

$$\sup_t \left\| \int_0^t e^{-(n-1)a\tau}\, \overline{\mathrm{B}}\left[v^n\right] d\tau \right\|_{L_\infty} \leqslant \sup_t \int_0^t \left\|\overline{\mathrm{B}}\left[v^n\right]\right\|_{L_\infty} d\tau \leqslant$$
$$\leqslant \sup_t \left\|\overline{\mathrm{B}}\left[v^n\right]\right\|_{L_\infty} \int_0^\infty e^{-(n-1)a\tau} d\tau \leqslant \frac{b}{(n-1)a}\|v\|^n_{L_\infty} \leqslant \frac{b}{(k-1)a}\|v\|^n_{L_\infty}$$

Следовательно,

$$\sup_t \|Mv\|_{L_\infty} \leqslant \|u_0\|_{L_\infty} + \frac{b}{(k-1)a} \sum_{n=k}^{\infty} |\mu_n| \rho^n$$

Докажем теперь аналогичное соотношение в пространстве H^2. С помощью лемм 2 и 4 получим:

$$\sup_t \left\| \int_0^t e^{-(n-1)a\tau}\, \overline{\mathrm{B}}\left[v^n\right] d\tau \right\|_{H^2} \leqslant c_3 \sup_t \left\| \int_0^t v^n(y,\tau)\, e^{-(n-1)a\tau} d\tau \right\|_{L_2}$$

С учетом леммы 5 продолжим преобразование:

$$\Rightarrow \sup_t \left\| \int_0^t e^{-(n-1)a\tau}\, \overline{\mathrm{B}}\left[v^n\right] d\tau \right\|_{H^2} \leqslant \frac{c_3}{(k-1)a} \sup_t \|\varphi\|^{n-1}_{L_\infty} \sup_t \|\varphi\|_{H^2}$$

Следовательно,

$$\sup_t \|Mv\|_{H^2} \leqslant \|u_0\|_{L_\infty} + \frac{c_3}{(k-1)a} \sum_{n=k}^{\infty} |\mu_n| \rho^n$$

Так как $\|\varphi\|_{L_\infty} \leqslant \|\varphi\|$ и $\|\varphi\|_{H^2} \leqslant \|\varphi\|$, то

$$\|Mv\| \leqslant \|u_0\| + \frac{1}{(k-1)a} \max(b,\ c_3) \sum_{n=k}^{\infty} |\mu_n| \rho^n \leqslant \rho$$

при достаточно малых ρ.

Таким образом, оператор M переводит шар X_ρ в себя.

Доказательство оценки $\|Mw - Mv\| \leqslant 1/2 \cdot \|w - v\| \ \forall w,\ v \in X$, означающей, что оператор M является сжимающим отображением, аналогично доказательству утверждения, что этот оператор переводит шар X_ρ в себя. Здесь следует принять во внимание неравенство $|w^n - v^n| \leqslant n\rho^{n-1}\,|w - v|$.

Итак, оператор M переводит шар X_ρ в себя и является сжимающим отображением. Значит, существует единственное решение уравнения $Mw = w$ из пространства X. **Теорема доказана.**

Теорема 2. *Существует такая окрестность нуля, что если $\|u_0\|$ находится в этой окрестности, то при $t \to \infty$ имеет место следующая асимптотика:*

$$u\,(x,t) = A\,(x)\,e^{-at} + O\left(e^{-kat}\right),$$

где

$$A\,(x) = u_0\,(x) + \sum_{n=k}^{\infty} \mu_n \int\limits_0^{\infty} e^{a\tau}\,\overline{\mathrm{B}}\,[u^n]\,d\tau$$

Таким образом, если начальные данные достаточно малы в смысле нормы пространства X, то главная составляющая асимптотики полностью определяется линейными членами уравнения, тогда как нелинейность влияет только на порядок стремления к нулю остаточного члена. В частности, если нелинейность тождественно равна нулю, то главный член асимптотики является точным решением: $u\,(x,t) = A\,(x)\,e^{-at} = u_0\,(x)\,e^{-at}$.

Докажем названную асимптотическую формулу.

Исходя из уравнения (3), представим v в следующем виде:

$$v\,(x,t) = u_0\,(x) + \sum_{n=k}^{\infty} \mu_n \int\limits_0^{\infty} e^{-(n-1)a\tau}\,\overline{\mathrm{B}}\,[v^n]\,d\tau - \sum_{n=k}^{\infty} \mu_n \int\limits_t^{\infty} e^{-(n-1)a\tau}\,\overline{\mathrm{B}}\,[v^n]\,d\tau$$

Обозначим сумму первых двух слагаемых через $A(x)$ и перейдем к первоначальной переменной u. Функция $A(\cdot)$ не зависит от времени. Обозначим через R третье слагаемое. Докажем, что $R = O\left(e^{-(k-1)at}\right)$. Заметим, что

$$R = \sum_{n=k}^{\infty} \mu_n \int\limits_t^{\infty} e^{-(n-1)a\tau} \int\limits_{R^N} B\,(x-y)\,v^n\,(y,\tau)\,dy d\tau =$$
$$= \sum_{n=k}^{\infty} \mu_n \int\limits_{R^N} B\,(x-y)\,dy \int\limits_t^{\infty} e^{-(n-1)a\tau} v^n\,(y,\tau)\,d\tau$$

Следовательно,

$$|R| = \left| \int\limits_t^{\infty} \int\limits_{R^N} B\,(x-y) \sum_{n=k}^{\infty} \mu_n e^{-(n-1)a\tau} v^n\,(y,\tau)\,d\tau dy \right| \leqslant$$
$$\leqslant \int\limits_t^{\infty} e^{-(k-1)a\tau} d\tau \int\limits_{R^N} |B\,(x-y)| \sum_{n=k}^{\infty} |\mu_n|\,|v\,(y,\tau)|^n\,dy \leqslant$$
$$\leqslant \int\limits_t^{\infty} e^{-(k-1)a\tau} d\tau \int\limits_{R^N} |B\,(x-y)|\,dy \sum_{n=k}^{\infty} |\mu_n|\,\rho^n =$$
$$= \frac{e^{-(k-1)at}}{(k-1)a} \cdot b \sum_{n=k}^{\infty} |\mu_n|\,\rho^n = O\left(e^{-(k-1)at}\right)$$

Итак,

$$v(x,t) = A(x) + O\left(e^{-(k-1)at}\right),$$

откуда

$$u(x,t) = v(x,t)\,e^{-at} = A(x)\,e^{-at} + O\left(e^{-kat}\right)$$

Теорема доказана.

Список литературы

[1] G. N. Watson. A treatise on the theory of Bessel functions. Cambridge University Press, Cambridge, England, 1944.

[2] N. Hayashi, E. Kaikina, P. Naumkin, I. Shishmarev. Asymptotics for Dissipative Nonlinear Equations. Springer-Verlag. 2006. 564 с.

[3] А. Г. Свешников, А. Б. Альшин, М. О. Корпусов. Нелинейный функциональный анализ и его приложения к уравнениям в частных производных. М., Научный мир, 2008.

УДК 517.951

ЕДИНСТВЕННОСТЬ ОБОБЩЕННЫХ РЕШЕНИЙ СМЕШАННЫХ ЗАДАЧ ДЛЯ ВОЛНОВОГО УРАВНЕНИЯ С НЕЛОКАЛЬНЫМИ ГРАНИЧНЫМИ УСЛОВИЯМИ

© 2009 г. А. В. Беликов

belikov.anton@gmail.com

Кафедра Общей математики

1 Постановка задачи

Рассмотрим описываемый волновым уравнением $u_{tt}(x,t) - u_{xx}(x,t) = 0$ и протекающий за промежуток времени $0 \le t \le T$ процесс колебаний струны, концами которой служат точки $x = 0$ и $x = l$.

В настоящей работе устанавливаются теоремы единственности обобщенных решений смешанных задач для волнового уравнения с нелокальными граничными условиями четырех типов с заданными либо начальными, либо финальными условиями.

Сначала будем рассматривать случай задания начальных условий, а затем рассмотрим случай задания финальных условий.

2 Основные определения

Обозначим через Q_T прямоугольник $Q_T = [0 \le x \le l] \times [0 \le t \le T]$, а символом $\widehat{W}_2^2(Q_T)$ класс функций двух переменных $u(x,t)$, определения которых будут даны ниже.

Определение 1. Прямоугольник Q_T будем называть основным прямоугольником.

Определение 2. Будем говорить, что функция двух переменных $u(x,t)$ принадлежит классу $\widehat{W}_2^2(Q_T)$, если сама функция $u(x,t)$ и все ее частные производные первого порядка непрерывны в замкнутом прямоугольнике $\overline{Q_T}$ и если у этой функции существуют в Q_T все обобщенные производные второго порядка, каждая из которых принадлежит классу $L_2[0 \le x \le l]$ при любом t из сегмента $0 \le t \le T$ и принадлежит классу $L_2[0 \le t \le T]$ при любом x из сегмента $0 \le x \le l$.

Пусть $\xi_1, \xi_2, \ldots, \xi_p$ - произвольные внутренние точки сегмента $[0,l]$, удовлетворяющие неравенствам

$$0 \le \xi_1 \le \xi_2 \le \xi_3 \le \ \ldots \le \xi_p \le l, \tag{1}$$

а $\alpha_1(t), \alpha_2(t), \ldots, \alpha_p(t)$ - произвольные ограниченные на сегменте $[0,T]$ функции, удовлетворяющие условиям, обеспечивающим существование обобщенных решений рассматриваемых ниже смешанных задач.

Определим для волнового уравнения

$$u_{tt}(x,t) - u_{xx}(x,t) = 0 \tag{2}$$

в прямоугольнике Q_T обобщенные из класса $\widehat{W}_2^2(Q_T)$ решения следующих четырех смешанных задач с нелокальными граничными условиями:

задачи 1 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения волнового уравнения (2) с начальными условиями

$$u(x,0) = \varphi(x), \ u_t(x,t) = \psi(x) \tag{3}$$

и граничными условиями

$$u(0,t)=\mu(t),\ u(l,t)=\sum_{k=1}^{p}\alpha_k(t)u(\xi_k,t); \tag{4}$$

задачи 2 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения волнового уравнения (2) с начальными условиями (3) и граничными условиями

$$u(0,t)=\mu(t),\ u_x(l,t)=\sum_{k=1}^{p}\alpha_k(t)u_x(\xi_k,t); \tag{5}$$

задачи 3 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения волнового уравнения (2) с начальными условиями (3) и граничными условиями

$$u_x(0,t)=\mu(t),\ u(l,t)=\sum_{k=1}^{p}\alpha_k(t)u(\xi_k,t); \tag{6}$$

задачи 4 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения волнового уравнения (2) с начальными условиями (3) и граничными условиями

$$u_x(0,t)=\mu(t),\ u_x(l,t)=\sum_{k=1}^{p}\alpha_k(t)u_x(\xi_k,t). \tag{7}$$

Замечание. Теоремы существования и единственности обобщенных из класса $\widehat{W}_2^1(Q_T)$ решений данных смешанных задач 1-4 рассматривались в работе [1].

Замечание. При замене каждой из сумм, стоящих в граничных условиях (4)-(7), заданной функцией задачи 1-4 превращаются в хорошо изученные (например, в [2]-[7]) задачи с локальными граничными условиями. В работах [2]-[7] рассматривалось функциональное пространство $\widehat{W}_2^1(Q_T)$, поэтому требовались интегральные тождества, которым должны удовлетворять обобщенные из класса $\widehat{W}_2^1(Q_T)$ решения смешанных задач 1-4, но в случае функционального пространства $\widehat{W}_2^2(Q_T)$ требуется удовлетворение волновому уравнению (2) почти всюду.

Определение 3. Обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанных задач 1-4 называется такая функция $u(x,t)$ из класса $\widehat{W}_2^2(Q_T)$, которая удовлетворяет почти всюду волновому уравнению (2), начальным условиям (3) и соответствующим граничным условиям (4)-(7).

Замечание. Из принадлежности решения $u(x,t)$ каждой из задач 1-4 классу $\widehat{W}_2^2(Q_T)$ вытекают точные условия гладкости, которым должны удовлетворять функции $\varphi(x)$ и $\psi(x)$ из начальных условий (3) и функция $\mu(x)$ из граничных условий (4)-(7): $\varphi(x)\in W_2^2[0,l]$, $\psi(x)\in W_2^1[0,l]$, $\mu(t)\in W_2^2[0,T]$ в задачах 1 и 2, $\mu(t)\in W_2^1[0,T]$ в задачах 3 и 4.

Замечание. Кроме того, для задач 1 и 2 должно выполняться равенство $\varphi(0)=\mu(0)$.

3 Вспомогательные задачи

Для вспомогательных целей рассмотрим в классе функций $\widehat{W}_2^2(Q_T)$ три смешанные задачи для волнового уравнения с заданными начальными условиями и с локальными граничными управлениями следующего вида:

задача 1: граничные управления представляют собой управления смещениями на двух концах струны;

задача 2: граничные управления представляют собой управления смещением на одном конце и упругой силой на другом конце струны;

задача 3: граничные управления представляют собой управления упругими силами на двух концах струны.

Замечание. Данные задачи в функциональном пространстве $\widehat{W}_2^1(Q_T)$ подробно рассмотрены в работах [2]-[7].

Рассмотрим данные задачи в исследуемом в данной работе функциональном пространстве $\widehat{W}_2^2(Q_T)$, докажем существование и единственность решений и найдем их явный аналитический вид в пространстве $\widehat{W}_2^2(Q_T)$.

3.1 Управление смещениями на двух концах струны

Рассмотрим в основном прямоугольнике Q_T задачу для волнового уравнения (2) с начальными условиями (3) и с граничными условиями

$$u(0,t)=\mu(t),\ u(l,t)=\nu(t). \tag{8}$$

При этом принадлежность обобщенного решения $u(x,t)$ классу $\widehat{W}_2^2(Q_T)$ позволяет точно указать требования гладкости, которым следует подчинить функции начальных и граничных условий:

$$\varphi(x)\in W_2^2[0,l],\ \psi(x)\in W_2^1[0,l],\ \mu(t)\in W_2^2[0,T],\ \nu(t)\in W_2^2[0,T]. \tag{9}$$

Определение 4. Обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения (2) с начальными условиями (3) и граничными условиями (8), подчиненными требованиям гладкости (9) назовем функцию $u(x,t)$ из класса $\widehat{W}_2^2(Q_T)$, удовлетворяющую почти всюду волновому уравнению (2), начальными условиям (3) и граничным условиям (8).

Замечание. Из рассуждений, проведенных в [11, Гл.2, §9], вытекает, что для любого $T>0$ может существовать только одно обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение определенной нами смешанной задачи.

Как и в работах [2]-[7] введем в рассмотрение функцию $\widetilde{u}(x,t)$, определив ее равенством:

$$\widetilde{u}(x,t)=\begin{cases}\frac{1}{2}[\varphi(x+t)+\varphi(x-t)+\int_{x-t}^{x+t}\psi(\xi)d\xi] & \text{в } \Delta_1,\\ \frac{1}{2}[\varphi(x+t)+\varphi(0)+\int_0^{x+t}\psi(\xi)d\xi] & \text{в } \Delta_2,\\ \frac{1}{2}[\varphi(l)+\varphi(x-t)+\int_{x-t}^{l}\psi(\xi)d\xi] & \text{в } \Delta_3,\\ \frac{1}{2}[\varphi(0)+\varphi(l)+\int_0^{l}\psi(\xi)d\xi]=C_0=const & \text{в } \Delta_4,\end{cases} \tag{10}$$

в котором Δ_1 - треугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$ и $t=0$, Δ_2 - треугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$ и $x=0$, Δ_3 - треугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$ и $x-l=0$, Δ_4 - пятиугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$, $x=0$ и $t-T=0$.

Утверждение 1. *При любом $T>l$, а также при выполнении условий $\varphi(0)=\psi(0)$, $\varphi(l)=-\psi(l)$, функция (10) является (единственным в силу работы [11]) обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения $\widetilde{u}_{tt}(x,t)-\widetilde{u}_{xx}(x,t)=0$ с начальными условиями $\widetilde{u}(x,0)=\varphi(x)$, $\widetilde{u}_t(x,0)=\psi(x)$ и с граничными условиями $\widetilde{u}(0,t)=\widetilde{\mu}(t)$, $\widetilde{u}(l,t)=\widetilde{\nu}(t)$, в которых*

$$\widetilde{\mu}(t)=\begin{cases}\frac{1}{2}[\varphi(t)+\varphi(0)+\int_0^{t}\psi(\xi)d\xi] & \text{при } 0\le t\le l,\\ C_0=\frac{1}{2}[\varphi(0)+\varphi(l)+\int_0^{l}\psi(\xi)d\xi] & \text{при } l\le t\le T,\end{cases} \tag{11}$$

$$\widetilde{\nu}(t)=\begin{cases}\frac{1}{2}[\varphi(l-t)+\varphi(l)+\int_{l-t}^{l}\psi(\xi)d\xi] & \text{при } 0\le t\le l,\\ C_0=\frac{1}{2}[\varphi(0)+\varphi(l)+\int_0^{l}\psi(\xi)d\xi] & \text{при } l\le t\le T.\end{cases} \tag{12}$$

Доказательство. Рассмотрим частные производные функции $\widetilde{u}(x,t)$ первого порядка:

$$\widetilde{u}_x(x,t)=\begin{cases}\frac{1}{2}[\varphi'(x+t)+\varphi'(x-t)-\psi(x+t)+\psi(x-t)] & \text{в } \Delta_1,\\ \frac{1}{2}[\varphi'(x+t)+\psi(x+t)] & \text{в } \Delta_2,\\ \frac{1}{2}[\varphi'(x-t)-\psi(x-t)] & \text{в } \Delta_3,\\ 0 & \text{в } \Delta_4,\end{cases} \tag{13}$$

$$\widetilde{u}_t(x,t)=\begin{cases}\frac{1}{2}[\varphi'(x+t)-\varphi'(x-t)+\psi(x+t)+\psi(x-t)] & \text{в } \Delta_1,\\ \frac{1}{2}[\varphi'(x+t)+\psi(x+t)] & \text{в } \Delta_2,\\ \frac{1}{2}[-\varphi'(x-t)+\psi(x-t)] & \text{в } \Delta_3,\\ 0 & \text{в } \Delta_4.\end{cases} \tag{14}$$

Принадлежность функции (10) классу $\widehat{W}_2^2(Q_T)$ вытекает из того, что эта функция в каждой из областей Δ_1, Δ_2, Δ_3 и Δ_4 представляет собой алгебраическую сумму функций от аргумента $x+t$ или $x-t$, имеющих суммируемые с квадратом все обобщенные производные второго порядка, и сохраняет непрерывность при переходе через границу любых двух из указанных четырех областей, а обобщенные производные функции (10), представленные формулами (13) и (14) в силу условий $\varphi'(0)=\psi(0)$, $\varphi'(l)=-\psi(l)$ также представляют собой алгебраические суммы функций от аргумента $x+t$ или $x-t$, имеющих суммируемую с квадратом обобщенную производную, и сохраняют непрерывность при переходе через границу любых двух из указанных четырех областей.

Также очевидно, что при выполнении условий данного утверждения функции $\widetilde{\mu}(t)$ и $\widetilde{\nu}(t)$ принадлежат классу $W_2^2[0,T]$.

Тривиально проверяется, что функция (10) удовлетворяет волновому уравнению $\widetilde{u}_{tt}(x,t)-\widetilde{u}_{xx}(x,t)=0$ почти всюду. Утверждение 1 доказано.

Опустив рассмотрение тривиального случая $l<T<2l$ мы в дальнейшем будем рассматривать произвольный промежуток времени T, удовлетворяющий неравенству $T\geq 2l$, представив указанное T в виде

$$T=2ln+\Delta, \tag{15}$$

где $n=1,2,3,\ldots$, а Δ - действительное число, удовлетворяющее неравенствам $0\leq\Delta\leq 2l$.

Пусть $u(x,t)$ - обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение рассматриваемой смешанной задачи для волнового уравнения (2) с начальными условиями (3) и с граничными условиями (8), а $\widetilde{u}(x,t)$ - функция (10). Тогда функция $\widehat{u}(x,t)=u(x,t)-\widetilde{u}(x,t)$ является единственным обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения $\widehat{u}_{tt}(x,t)-\widehat{u}_{xx}(x,t)=0$ с нулевыми начальными условиями $\widehat{u}(x,0)=0$, $\widehat{u}_t(x,0)=0$ и с граничными условиями $\widehat{u}(0,t)=\widehat{\mu}(t)=\mu(t)-\widetilde{\mu}(t)$, $\widehat{u}(l,t)=\widehat{\nu}(t)=\nu(t)-\widetilde{\nu}(t)$. Заметим при этом, что из равенств $\mu(0)=\varphi(0)$, $\widetilde{\mu}(0)=\varphi(0)$, $\nu(0)=\varphi(l)$, $\widetilde{\nu}(0)=\varphi(l)$ вытекает, что $\widehat{\mu}(0)=0$, $\widehat{\nu}(0)=0$.

Утверждение 2. *Для любого T, удовлетворяющего неравенству $T\leq 2l(n+1)$, где $n=1,2,3,\ldots$ (и, в частности, для T, удовлетворяющего условию (15) функция $\widehat{u}(x,t)$ определяется равенством*

$$\widehat{u}(x,t)=\sum_{k=0}^{n}\underline{\widehat{\mu}}(t-x-2kl)-\sum_{k=1}^{n+1}\underline{\widehat{\mu}}(t+x-2kl)+ \\ +\sum_{k=0}^{n}\underline{\widehat{\nu}}(t+x-2kl-l)-\sum_{k=1}^{n-1}\underline{\widehat{\nu}}(t-x-2kl+l), \tag{16}$$

в котором символы $\underline{\widehat{\mu}}(t)$ и $\underline{\widehat{\nu}}(t)$ обозначают функции, совпадающие с $\widehat{\mu}(t)$ и $\widehat{\nu}(t)$ соответственно при $t\geq 0$ и равные нулю при $t<0$.

Доказательство. Сразу же заметим, что функция (16) принадлежит классу $\widehat{W}_2^2(Q_T)$ в силу того, что она представляет собой алгебраическую сумму конечного числа функций от аргумента $t+x$ или $t-x$, каждая из которых принадлежит классу $W_2^2(-\infty,T]$.

Функция (16) удовлетворяет почти всюду волновому уравнению $\widehat{u}_{tt}(x,t)-\widehat{u}_{xx}(x,t)=0$, нулевым начальным условиям $\widehat{u}(x,0)=0$, $\widehat{u}_t(x,0)=0$ и граничным условиям $\widehat{u}(0,t)=\widehat{\mu}(t)=\mu(t)-\widetilde{\mu}(t)$, $\widehat{u}(l,t)=\widehat{\nu}(t)=\nu(t)-\widetilde{\nu}(t)$. Значит, данная функция является единственным решением описанной выше задачи и имеет указанный аналитический вид. Утверждение 2 доказано.

3.2 Управление смещением на одном конце и упругой силой на другом конце струны

Рассмотрим в основном прямоугольнике Q_T задачу для волнового уравнения (2) с начальными условиями (3) и с граничными условиями

$$u_x(0,t) = \mu(t),\ u(l,t) = \nu(t). \tag{17}$$

При этом принадлежность обобщенного решения $u(x,t)$ классу $\widehat{W}_2^2(Q_T)$ позволяет точно указать требования гладкости, которым следует подчинить функции начальных и граничных условий:

$$\varphi(x) \in W_2^2[0,l],\ \psi(x) \in W_2^1[0,l],\ \mu(t) \in W_2^1[0,T],\ \nu(t) \in W_2^2[0,T]. \tag{18}$$

Определение 5. Обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения (2) с начальными условиями (3) и граничными условиями (17), подчиненными требованиям гладкости (18) назовем функцию $u(x,t)$ из класса $\widehat{W}_2^2(Q_T)$, удовлетворяющую почти всюду волновому уравнению (2), начальными условиям (3) и граничным условиям (17).

Замечание. Из рассуждений, проведенных в [11, Гл.2, §9], вытекает, что для любого $T > 0$ может существовать только одно обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение определенной нами смешанной задачи.

Как и в случае задачи с граничными управлениями смещениями на двух концах струны, рассмотренной выше, снова введем функцию $\widetilde{u}(x,t)$, определяемую равенством (10), в котором Δ_1 - треугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$ и $t=0$, Δ_2 - треугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$ и $x=0$, Δ_3 - треугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$ и $x-l=0$, Δ_4 - пятиугольник, ограниченный отрезками прямых $x-t=0$, $x+t-l=0$, $x=0$ и $t-T=0$.

Утверждение 3. *При любом $T > l$, а также при выполнении условий $\varphi(0) = \psi(0)$, $\varphi(l) = -\psi(l)$, функция (10) является (единственным в силу работы [11]) обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения $\widetilde{u}_{tt}(x,t) - \widetilde{u}_{xx}(x,t) = 0$ с начальными условиями $\widetilde{u}(x,0) = \varphi(x)$, $\widetilde{u}_t(x,0) = \psi(x)$ и с граничными условиями $\widetilde{u}(0,t) = \widetilde{\mu}(t)$, $\widetilde{u}(l,t) = \widetilde{\nu}(t)$, в которых*

$$\widetilde{\mu}(t) = \begin{cases} \frac{1}{2}[\varphi'(t) + \psi(t)] & \text{при } 0 \le t \le l, \\ 0 & \text{при } l \le t \le T, \end{cases} \tag{19}$$

$$\widetilde{\nu}(t) = \begin{cases} \frac{1}{2}[\varphi(l-t) + \varphi(l) + \int_{l-t}^{l} \psi(\xi)d\xi] & \text{при } 0 \le t \le l, \\ C_0 = \frac{1}{2}[\varphi(0) + \varphi(l) + \int_0^l \psi(\xi)d\xi] & \text{при } l \le t \le T. \end{cases} \tag{20}$$

Доказательство. Доказательство данного утверждения полностью аналогично доказательству Утверждения 1 данной работы. Утверждение 3 доказано.

Пусть $u(x,t)$ - обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение рассматриваемой смешанной задачи для волнового уравнения (2) с начальными условиями (3) и с граничными условиями (17), а $\widetilde{u}(x,t)$ - функция (10). Тогда функция $\widehat{u}(x,t) = u(x,t) - \widetilde{u}(x,t)$ является единственным обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения $\widehat{u}_{tt}(x,t) - \widehat{u}_{xx}(x,t) = 0$ с нулевыми начальными условиями $\widehat{u}(x,0) = 0$, $\widehat{u}_t(x,0) = 0$ и с граничными условиями $\widehat{u}_x(0,t) = \widehat{\mu}(t) = \mu(t) - \widetilde{\mu}(t)$, $\widehat{u}(l,t) = \widehat{\nu}(t) = \nu(t) - \widetilde{\nu}(t)$. Заметим при этом, что из равенства $\nu(0) = \varphi(l)$, $\widetilde{\nu}(0) = \varphi(l)$ вытекает, что $\widehat{\nu}(0) = 0$.

Утверждение 4. *Для любого T, удовлетворяющего неравенству $T \le 4l(n+1)$, где $n = 1,2,3,\ldots$ (и, в частности, для T, удовлетворяющего условию (15) функция $\widehat{u}(x,t)$ определяется равенством*

$$\widehat{u}(x,t) = -\sum_{k=0}^{2n+1}(-1)^k \int_0^{t-x-2kl} \underline{\widehat{\mu}}(\tau)d\tau - \sum_{k=1}^{2n+2}(-1)^k \int_0^{t+x-2kl} \underline{\widehat{\mu}}(\tau)d\tau + \\ + \sum_{k=0}^{2n+1}(-1)^k \underline{\widehat{\nu}}(t+x-2kl-l) - \sum_{k=1}^{2n+2}(-1)^k \underline{\widehat{\nu}}(t-x-2kl+l), \tag{21}$$

в котором символы $\underline{\widehat{\mu}}(t)$ *и* $\underline{\widehat{\nu}}(t)$ *обозначают функции, совпадающие с* $\widehat{\mu}(t)$ *и* $\widehat{\nu}(t)$ *соответственно при* $t \geq 0$ *и равные нулю при* $t < 0$.

Доказательство. Сразу же заметим, что функция (21) принадлежит классу $\widehat{W}_2^2(Q_T)$ в силу того, что она представляет собой алгебраическую сумму конечного числа функций от аргумента $t + x$ или $t - x$, каждая из которых принадлежит классу $W_2^2(-\infty, T]$.

Функция (21) удовлетворяет почти всюду волновому уравнению $\widehat{u}_{tt}(x,t) - \widehat{u}_{xx}(x,t) = 0$, нулевым начальным условиям $\widehat{u}(x,0) = 0,\ \widehat{u}_t(x,0) = 0$ и граничным условиям $\widehat{u}_x(0,t) = \widehat{\mu}(t) = \mu(t) - \widetilde{\mu}(t),\ \widehat{u}(l,t) = \widehat{\nu}(t) = \nu(t) - \widetilde{\nu}(t)$. Значит, данная функция является единственным решением описанной выше задачи и имеет указанный аналитический вид. Утверждение 4 доказано.

3.3 Управление упругими силами на двух концах струны

Рассмотрим в основном прямоугольнике Q_T задачу для волнового уравнения (2) с начальными условиями (3) и с граничными условиями

$$u_x(0,t) = \mu(t),\ u_x(l,t) = \nu(t). \tag{22}$$

При этом принадлежность обобщенного решения $u(x,t)$ классу $\widehat{W}_2^2(Q_T)$ позволяет точно указать требования гладкости, которым следует подчинить функции начальных и граничных условий:

$$\varphi(x) \in W_2^2[0,l],\ \psi(x) \in W_2^1[0,l],\ \mu(t) \in W_2^1[0,T],\ \nu(t) \in W_2^1[0,T]. \tag{23}$$

Определение 6. Обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения (2) с начальными условиями (3) и граничными условиями (22), подчиненными требованиям гладкости (23) назовем функцию $u(x,t)$ из класса $\widehat{W}_2^2(Q_T)$, удовлетворяющую почти всюду волновому уравнению (2), начальными условиям (3) и граничным условиям (22).

Замечание. Из рассуждений, проведенных в [11, Гл.2, §9], вытекает, что для любого $T > 0$ может существовать только одно обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение определенной нами смешанной задачи.

Как и в случае задачи с граничными управлениями смещениями на двух концах струны и задачи с граничными управлениями смещением на одном конце и упругой силой на другом конце струны, рассмотренных выше, снова введем функцию $\widetilde{u}(x,t)$, определяемую равенством (10), в котором Δ_1 - треугольник, ограниченный отрезками прямых $x - t = 0$, $x + t - l = 0$ и $t = 0$, Δ_2 - треугольник, ограниченный отрезками прямых $x - t = 0$, $x + t - l = 0$ и $x = 0$, Δ_3 - треугольник, ограниченный отрезками прямых $x - t = 0$, $x + t - l = 0$ и $x - l = 0$, Δ_4 - пятиугольник, ограниченный отрезками прямых $x - t = 0$, $x + t - l = 0$, $x = 0$ и $t - T = 0$.

Утверждение 5. *При любом* $T > l$, *а также при выполнении условий* $\varphi(0) = \psi(0),\ \varphi(l) = -\psi(l)$, *функция (10) является (единственным в силу работы [11]) обобщенным из класса* $\widehat{W}_2^2(Q_T)$ *решением смешанной задачи для волнового уравнения* $\widetilde{u}_{tt}(x,t) - \widetilde{u}_{xx}(x,t) = 0$ *с начальными условиями* $\widetilde{u}(x,0) = \varphi(x),\ \widetilde{u}_t(x,0) = \psi(x)$ *и с граничными условиями* $\widetilde{u}_x(0,t) = \widetilde{\mu}(t),\ \widetilde{u}_x(l,t) = \widetilde{\nu}(t)$, *в которых*

$$\widetilde{\mu}(t) = \begin{cases} \frac{1}{2}[\varphi'(t) + \psi(t)] & \text{при } 0 \leq t \leq l, \\ 0 & \text{при } l \leq t \leq T, \end{cases} \tag{24}$$

$$\widetilde{\nu}(t) = \begin{cases} \frac{1}{2}[\varphi'(l-t) - \psi(l-t)] & \text{при } 0 \leq t \leq l, \\ 0 & \text{при } l \leq t \leq T. \end{cases} \tag{25}$$

Доказательство. Доказательство данного утверждения полностью аналогично доказательству Утверждения 1 данной работы. Утверждение 5 доказано.

Пусть $u(x,t)$ - обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение рассматриваемой смешанной задачи для волнового уравнения (2) с начальными условиями (3) и с граничными условиями (22), а $\widetilde{u}(x,t)$ - функция (10). Тогда функция $\widehat{u}(x,t) = u(x,t) - \widetilde{u}(x,t)$ является единственным обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения $\widehat{u}_{tt}(x,t) - \widehat{u}_{xx}(x,t) = 0$ с нулевыми начальными условиями $\widehat{u}(x,0) = 0,\ \widehat{u}_t(x,0) = 0$ и с граничными условиями $\widehat{u}_x(0,t) = \widehat{\mu}(t) = \mu(t) - \widetilde{\mu}(t),\ \widehat{u}_x(l,t) = \widehat{\nu}(t) = \nu(t) - \widetilde{\nu}(t)$.

Утверждение 6. *Для любого T, удовлетворяющего неравенству $T \le 2l(n+1)$, где $n = 1, 2, 3, \ldots$ (и, в частности, для T, удовлетворяющего условию (15) функция $\widehat{u}(x,t)$ определяется равенством*

$$\widehat{u}(x,t) = -\sum_{k=0}^{n}\int_0^{t-x-2kl}\underline{\widehat{\mu}}(\tau)d\tau - \sum_{k=1}^{n+1}\int_0^{t+x-2kl}\underline{\widehat{\mu}}(\tau)d\tau + \\ + \sum_{k=0}^{n}\int_0^{t+x-2kl-l}\underline{\widehat{\nu}}(\tau)d\tau + \sum_{k=1}^{n+1}\int_0^{t-x-2kl+l}\underline{\widehat{\nu}}(\tau)d\tau, \quad (26)$$

в котором символы $\underline{\widehat{\mu}}(t)$ и $\underline{\widehat{\nu}}(t)$ обозначают функции, совпадающие с $\widehat{\mu}(t)$ и $\widehat{\nu}(t)$ соответственно при $t \ge 0$ и равные нулю при $t < 0$.

Доказательство. Сразу же заметим, что функция (26) принадлежит классу $\widehat{W}_2^2(Q_T)$ в силу того, что она представляет собой алгебраическую сумму конечного числа функций от аргумента $t+x$ или $t-x$, каждая из которых принадлежит классу $W_2^2(-\infty, T]$.

Функция (26) удовлетворяет почти всюду волновому уравнению $\widehat{u}_{tt}(x,t) - \widehat{u}_{xx}(x,t) = 0$, нулевым начальным условиям $\widehat{u}(x,0) = 0$, $\widehat{u}_t(x,0) = 0$ и граничным условиям $\widehat{u}_x(0,t) = \widehat{\mu}(t) = \mu(t) - \widetilde{\mu}(t)$, $\widehat{u}_x(l,t) = \widehat{\nu}(t) = \nu(t) - \widetilde{\nu}(t)$. Значит, данная функция является единственным решением описанной выше задачи и имеет указанный аналитический вид. Утверждение 6 доказано.

3.4 Замечания

При решении рассмотренных выше трех задач можно рассматривать любое $T > 0$. Поэтому, говоря об аналитическом виде функции $\widehat{u}(x,t)$ в каждой из трех задач, можно в равенствах (16), (21) и (26) заменить в каждой сумме верхний предел суммирования на бесконечный, так как в силу определения функций $\underline{\widehat{\mu}}(t)$ и $\underline{\widehat{\nu}}(t)$ равенства сохранятся.

Окончательно имеем для **задачи с граничным управлением смещениями на двух концах струны:**

$$\widehat{u}(x,t) = \sum_{k=0}^{\infty}\underline{\widehat{\mu}}(t-x-2kl) - \sum_{k=1}^{\infty}\underline{\widehat{\mu}}(t+x-2kl) + \\ + \sum_{k=0}^{\infty}\underline{\widehat{\nu}}(t+x-2kl-l) - \sum_{k=1}^{\infty}\underline{\widehat{\nu}}(t-x-2kl+l), \quad (27)$$

для **задачи с граничным управлением смещением на одном конце и упругой силой на другом конце струны:**

$$\widehat{u}(x,t) = -\sum_{k=0}^{\infty}(-1)^k\int_0^{t-x-2kl}\underline{\widehat{\mu}}(\tau)d\tau - \sum_{k=1}^{\infty}(-1)^k\int_0^{t+x-2kl}\underline{\widehat{\mu}}(\tau)d\tau + \\ + \sum_{k=0}^{\infty}(-1)^k\underline{\widehat{\nu}}(t+x-2kl-l) - \sum_{k=1}^{\infty}(-1)^k\underline{\widehat{\nu}}(t-x-2kl+l), \quad (28)$$

для **задачи с граничным управлением упругими силами на двух концах струны:**

$$\widehat{u}(x,t) = -\sum_{k=0}^{\infty}\int_0^{t-x-2kl}\underline{\widehat{\mu}}(\tau)d\tau - \sum_{k=1}^{\infty}\int_0^{t+x-2kl}\underline{\widehat{\mu}}(\tau)d\tau + \\ + \sum_{k=0}^{\infty}\int_0^{t+x-2kl-l}\underline{\widehat{\nu}}(\tau)d\tau + \sum_{k=1}^{\infty}\int_0^{t-x-2kl+l}\underline{\widehat{\nu}}(\tau)d\tau, \quad (29)$$

где символы $\underline{\widehat{\mu}}(t)$ и $\underline{\widehat{\nu}}(t)$ обозначают функции, совпадающие с $\widehat{\mu}(t)$ и $\widehat{\nu}(t)$ соответственно при $t \ge 0$ и равные нулю при $t < 0$.

4 Основная теорема

Перейдем теперь к формулировке и доказательству основного результата настоящей работы.

Основная теорема. *Для любого $T > 0$ может существовать только одно обобщенное из класса $\widehat{W}_2^2(Q_T)$ решение каждой из смешанных задач 1-4 с нелокальными граничными условиями (4)-(7).*

Доказательство основной теоремы для всех задач 1-4 производится по аналогичной схеме, и поэтому мы особенно подробно остановимся только на доказательстве основной теоремы для смешанных задач 1 и 2.

Предположив, что существует два обобщенных из класса $\widehat{W}_2^2(Q_T)$ решения $u_1(x,t)$ и $u_2(x,t)$ смешанной задачи 1, мы получим, что их разность $u(x,t) = u_1(x,t) - u_2(x,t)$ является обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения (2) с нулевыми начальными условиями $u(x,0) = 0,\ u_t(x,0) = 0$ и с граничными условиями

$$u(0,t) = 0,\ u(l,t) = \sum_{k=1}^{p} \alpha_k(t) u(\xi_k, t). \tag{30}$$

Так как указанное решение $u(x,t)$ принадлежит классу $\widehat{W}_2^2(Q_T)$, то по определению этого класса оно имеет при $x = l$ след $u(l,t)$, принадлежащий классу $W_2^2[0,T]$. Этот след мы обозначим символом $\widehat{\nu}(t)$ и заметим, что указанное решение $u(x,t)$ волнового уравнения (2) с нулевыми начальными условиями $u(x,0) = 0,\ u_t(x,0) = 0$ и с нелокальными граничными условиями (30) одновременно является обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением волнового уравнения (2) с нулевыми начальными условиями $u(x,0) = 0,\ u_t(x,0) = 0$ и с локальными граничными условиями

$$u(0,t) = 0,\ u(l,t) = \sum_{k=1}^{p} \alpha_k(t) u(\xi_k, t), \tag{31}$$

т.е. является хорошо изученной задачей, рассмотренной выше в настоящей работе, имеющей для любого $T > 0$ единственное из класса $\widehat{W}_2^2(Q_T)$ обобщенное решение, определяемое равенством

$$\widehat{u}(x,t) = \sum_{m=0}^{\infty} \underline{\widehat{\nu}}(t + x - 2lm - l) - \sum_{m=1}^{\infty} \underline{\widehat{\nu}}(t - x - 2lm + l), \tag{32}$$

в котором символ $\underline{\widehat{\nu}}(t)$ обозначает функцию, совпадающую с $\widehat{\nu}(t)$ соответственно при $t \geq 0$ и равную нулю при $t < 0$.

Докажем, что $\widehat{\nu}(t) \equiv 0$ для всех $t \geq 0$, чем и будет завершено доказательство основной теоремы для задачи 1.

Сначала докажем, что $\widehat{\nu}(t) \equiv 0$ при $0 \leq t \leq 2l$. Переписав соотношение (32) в виде

$$u(x,t) = \underline{\widehat{\nu}}(t + x - l) - \underline{\widehat{\nu}}(t - x - l) + \\ + \left(\sum_{m=1}^{\infty} \underline{\widehat{\nu}}(t + x - l - 2lm) - \sum_{m=2}^{\infty} \underline{\widehat{\nu}}(t - x + l - 2lm) \right), \tag{33}$$

заметим, что при $0 \leq t \leq 2l$ выражение, стоящее в (33) в круглых скобках равно нулю, ибо при $0 \leq t \leq 2l$ аргументы у всех стоящих в круглых скобках функций $\underline{\widehat{\nu}}$ являются неположительными.

Отсюда следует, что при $0 \leq t \leq 2l$ функция $u(x,t)$ определяется равенством

$$u(x,t) = \underline{\widehat{\nu}}(t + x - l) - \underline{\widehat{\nu}}(t - x - l). \tag{34}$$

Заметим теперь, что функция (34) должна при $0 \leq t \leq 2l$ удовлетворять второму (нелокальному) условию (30), и поскольку при $0 \leq t \leq 2l$ справедливы равенства

$$u(l,t) = \underline{\widehat{\nu}}(t) - \underline{\widehat{\nu}}(t - 2l) = \widehat{\nu}(t),\ u(\xi_k, t) = \underline{\widehat{\nu}}(t + \xi_k - l) - \underline{\widehat{\nu}}(t - \xi_k - l), \tag{35}$$

то второе условие (30) принимает вид

$$\widehat{\nu}(t)=\sum_{k=1}^{p}\alpha_k(t)[\widehat{\underline{\nu}}(t+\xi_k-l)-\widehat{\underline{\nu}}(t-\xi_k-l)]. \tag{36}$$

Так как числа $\xi_1,\xi_2,\ldots,\xi_p$ удовлетворяют неравенствам (1) и наибольшее из этих чисел ξ_p строго меньше l, то $l-\xi_p>0$, и поэтому при $0\le t\le l-\xi_p$ аргументы всех функций $\widehat{\underline{\nu}}$, стоящих в правой части (36), являются неположительными, т.е. при $0\le t\le l-\xi_p$ правая (а потому и левая) часть (36) равна нулю.

Итак, $\widehat{\nu}(t)=0$ при $0\le t\le l-\xi_p$, а из этого и из того же равенства (36) вытекает, что правая (а потому и левая) часть (36) равна нулю при $0\le t\le 2(l-\xi_p)$, ибо при $0\le t\le 2(l-\xi_p)$ аргументы всех функций $\widehat{\underline{\nu}}$, стоящих в правой части (36), не превосходят числа $l-\xi_p$.

Из того, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2(l-\xi_p)$, и из того же равенства (36) получим, что $\widehat{\underline{\nu}}(t)=0$ при $0\le t\le 3(l-\xi_p)$. Продолжая аналогичные рассуждения далее, мы после s шагов получим, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le s(l-\xi_p)$. Если обозначить через s наименьший из номеров, для которого $s(l-\xi_p)\ge 2l$, то после s шагов мы получим, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2l$.

Доказательство того, что $\widehat{\nu}(t)\equiv 0$ для всех положительных значений t, проведем методом математической индукции. Достаточно, предположив, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2ln$ и номера $n\ge 1$, доказать, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2l(n+1)$.

Если $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2ln$, то для любого t, удовлетворяющего условию $t\le 2l(n+1)$, выражение, стоящее в (33) в круглых скобках, равно нулю (ибо при $t\le 2l(n+1)$ аргументы всех функций $\widehat{\underline{\nu}}$, стоящих в (33) в круглых скобках, не превосходят числа $2ln$).

Итак, при условии, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2ln$, для любого t, удовлетворяющего условию $t\le 2l(n+1)$, функция $u(x,t)$ определяется равенством (34). Снова, учитывая, что

$$u(l,t)=\widehat{\underline{\nu}}(t)-\widehat{\underline{\nu}}(t-2l)=\widehat{\nu}(t),\ u(\xi_k,t)=\sum_{k=1}^{p}\alpha_k(t)[\widehat{\underline{\nu}}(t+\xi_k-l)-\widehat{\underline{\nu}}(t-\xi_k-l)], \tag{37}$$

мы получаем из второго равенства (31), что при $2ln\le t\le 2l(n+1)$ справедливо соотношение (36).

Из (36) заключаем, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2ln+(l-\xi_p)$, ибо для таких значений t аргументы всех функций $\widehat{\underline{\nu}}$, стоящих в правой части (36), не превосходят числа $2ln$.

После этого из того же соотношения (36) заключаем, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2ln+2(l-\xi_p)$, ибо для таких значений t аргументы всех функций $\widehat{\underline{\nu}}$, стоящих в правой части (36), не превосходят числа $2ln+(l-\xi_p)$.

Продолжая аналогичные рассуждения далее, после s шагов мы получаем, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2ln+s(l-\xi_p)$.

Если в качестве номера s взять наименьший из номеров, удовлетворяющих неравенству $s(l-\xi_p)\ge 2l$, то мы получим, что $\widehat{\nu}(t)\equiv 0$ при $0\le t\le 2l(n+1)$.

Тем самым доказательство основной теоремы для смешанной задачи 1 завершено.

Перейдем к доказательству основной теоремы для смешанной задачи 2. Предполагая, что эта задача имеет два обобщенных из класса $\widehat{W}_2^2(Q_T)$ решения и обозначая из разность через $u(x,t)$, замечаем, что по определению класса $\widehat{W}_2^2(Q_T)$ производная $u_x(x,t)$ этой разности имеет при $x=l$ след $u_x(l,t)$, принадлежащий классу $W_2^1[0,T]$. Этот след мы обозначим символом $\widehat{\nu}(t)$ и учтем, что рассматриваемая разность $u(x,t)$ является обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанной задачи для волнового уравнения (2) с нулевыми начальными условиями и с локальными граничными условиями

$$u(0,t)=0,\ u_x(l,t)=\widehat{\nu}(t). \tag{38}$$

т.е. является хорошо изученной задачей, рассмотренной выше в настоящей работе, имеющей единственное в классе $\widehat{W}_2^2(Q_T)$ обобщенное решение. Производная этого единственного решения $u_x(x,t)$ определяется равенством

$$u_x(x,t)=\sum_{m=0}^{\infty}(-1)^m\widehat{\underline{\nu}}(t+x-l-2lm)-\sum_{m=1}^{\infty}(-1)^m\widehat{\underline{\nu}}(t-x+l-2lm), \tag{39}$$

в котором символ $\widehat{\underline{\nu}}(t)$ обозначает функцию, совпадающую с $\widehat{\nu}(t)$ соответственно при $t \geq 0$ и равную нулю при $t < 0$.

Следует отметить, что в отличие от (32) соотношение (39) является равенством элементов класса $W_2^1(0,T)$.

Докажем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1(0,T)$, чем и будет завершено доказательство основной теоремы для смешанной задачи 2.

Сначала докажем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2l]$. Переписав (39) в виде

$$u_x(x,t) = \widehat{\underline{\nu}}(t+x-l) + \widehat{\underline{\nu}}(t-x-l) + \\ + \left(\sum_{m=1}^{\infty} (-1)^m \widehat{\underline{\nu}}(t+x-l-2lm) - \sum_{m=2}^{\infty} (-1)^m \widehat{\underline{\nu}}(t-x+l-2lm) \right), \quad (40)$$

заметим, что выражение, стоящее в (40) в круглых скобках, является нулевым элементом класса $W_2^1[0,2l]$, ибо при $t \in [0,2l]$ аргументы у всех стоящих в круглых скобках функций $\widehat{\underline{\nu}}$ являются неположительными. Итак, при $t \in [0,2l]$ функция $u_x(x,t)$ определяется равенством

$$u_x(x,t) = \widehat{\underline{\nu}}(t+x-l) + \widehat{\underline{\nu}}(t-x-l). \quad (41)$$

Заметим теперь, что разность $u(x,t)$ двух обобщенных решений смешанной задачи 2 обязана при $t \in [0,2l]$ удовлетворять второму (нелокальному) условию (5), т.е. условию

$$u_x(l,t) = \sum_{k=1}^{p} \alpha_k(t) u_x(\xi_k, t). \quad (42)$$

Поскольку в силу (41) в $W_2^1[0,2l]$ справедливы равенства

$$u_x(l,t) = \widehat{\underline{\nu}}(t) + \widehat{\underline{\nu}}(t-2l) = \widehat{\nu}(t), \ u_x(\xi_k,t) = \widehat{\underline{\nu}}(t+\xi_k-l) + \widehat{\underline{\nu}}(t-\xi_k-l), \quad (43)$$

то условие (42) принимает вид следующего равенства элементов класса $W_2^1[0,2l]$:

$$\widehat{\nu}(t) = \sum_{k=1}^{p} \alpha_k(t) [\widehat{\underline{\nu}}(t+\xi_k-l) + \widehat{\underline{\nu}}(t-\xi_k-l)]. \quad (44)$$

Из равенства (44) сначала заключаем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,l-\xi_p]$, ибо при $t \in [0,l-\xi_p]$ аргументы всех функций $\widehat{\underline{\nu}}$, стоящих в правой части (44), являются неположительными. После этого из того же равенства (44) заключаем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2(l-\xi_p)]$, ибо при $t \in [0,2(l-\xi_p)]$ аргументы всех функций $\widehat{\nu}(t)$, стоящих в правой части (44), не превосходят числа $l-\xi_p$.

Продолжая аналогичные рассуждения далее, после достаточного числа s шагов мы получаем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2l]$.

Доказательство того, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2ln]$ при любом натуральном n, проводится методом математической индукции. Достаточно, предположив, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2ln]$ при $n \geq 1$, доказать, что при этом $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2l(n+1)]$.

Сначала заметим, что если $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2ln]$, то при любом t, удовлетворяющем условию $t \leq 2l(n+1)$, обращаются в нуль все слагаемые, стоящие в (40) в круглых скобках, и $\widehat{\nu}(t)$ определяется равенством (41). Учитывая далее, что при условии, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2ln]$ и при всех $t \leq 2l(n+1)$ остаются справедливыми соотношения (42) и (43), мы придем к справедливому для элементов класса $W_2^1[0,2l(n+1)]$ соотношению (44).

Из этого соотношения сначала заключаем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0,2ln+(l-\xi_p)]$, ибо при $t \in [0,2ln+(l-\xi_p)]$ аргументы всех стоящих в правой части (44)

функций $\widehat{\nu}(t)$ не превосходят числа $2ln$. После этого из того же соотношения (44) заключаем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0, 2ln+2(l-\xi_p)]$, ибо при $t \in [0, 2ln+2(l-\xi_p)]$ аргументы всех стоящих в правой части (44) функций $\widehat{\nu}(t)$ не превосходят числа $2ln+(l-\xi_p)$.

Продолжая аналогичные рассуждения далее, после достаточного числа s шагов мы получаем, что $\widehat{\nu}(t)$ является нулевым элементом класса $W_2^1[0, 2l(n+1)]$, что и завершает доказательство основной теоремы для смешанной задачи 2.

Доказательство основной теоремы для смешанных задач 3 и 4 является почти дословным повторением приведенного нами ее доказательства для смешанных задач 1 и 2.

Проблема единственности обобщенных решений смешанных задач 3 и 4 по изложенной нами схеме сводится к проблеме единственности обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения рассмотренных выше задач для волнового уравнения (2) с нулевыми начальными условиями и с локальными граничными условиями вида $u_x(0,t) = 0,\ u(l,t) = \widehat{\nu}(t)$, либо вида $u_x(0,t) = 0,\ u_x(l,t) = \widehat{\nu}(t)$.

Мы ограничимся лишь указаниями о том, что в случае смешанной задачи 3 для разности $u(x,t)$ двух ее решений вместо (32) получается равенство

$$\widehat{u}(x,t) = \sum_{m=0}^{\infty}(-1)^m\widehat{\underline{\nu}}(t+x-2lm-l) - \sum_{m=1}^{\infty}(-1)^m\widehat{\underline{\nu}}(t-x-2lm+l) \tag{45}$$

и вместо (36) - соотношение

$$\widehat{\nu}(t) = \sum_{k=1}^{p}\alpha_k(t)[\widehat{\underline{\nu}}(t+\xi_k-l) + \widehat{\underline{\nu}}(t-\xi_k-l)], \tag{46}$$

а в случае смешанной задачи 4 для производной $u_x(x,t)$ разности $u(x,t)$ двух ее решений вместо (39) получается равенство

$$u_x(x,t) = \sum_{m=0}^{\infty}\widehat{\underline{\nu}}(t+x-l-2lm) - \sum_{m=1}^{\infty}\widehat{\underline{\nu}}(t-x+l-2lm) \tag{47}$$

и вместо (44) - соотношение

$$\widehat{\nu}(t) = \sum_{k=1}^{p}\alpha_k(t)[\widehat{\underline{\nu}}(t+\xi_k-l) - \widehat{\underline{\nu}}(t-\xi_k-l)]. \tag{48}$$

Сделанные нами замечания завершают доказательство основной теоремы. Основная теорема доказана.

5 Случай задания финальных условий

Перейдем теперь к рассмотрению смешанных задач с нелокальными граничными условиями вида (4)-(7) в случае, когда вместо начальных условий задаются финальные условия.

Для обозначения решения смешанной задачи с финальными условиями будем использовать символ $\check{u}(x,t)$.

Все рассматриваемые ниже функции $\check{u}(x,t)$ являются обобщенными из класса $\widehat{W}_2^2(Q_T)$ решениями волнового уравнения

$$\check{u}_{tt}(x,t) - \check{u}_{xx}(x,t) = 0 \tag{49}$$

с заданными финальными условиями

$$\check{u}(x,T) = \widehat{\varphi}(x),\ \check{u}_t(x,T) = \widehat{\psi}(x). \tag{50}$$

Аналогами задач 1-4 с начальными условиями (3) являются следующие четыре задачи с финальными условиями:

задача 5 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения $\check{u}(x,t)$ смешанной задачи для волнового уравнения (49) с финальными условиями (50) и с граничными условиями

$$\check{u}(0,t) = \mu(t),\ \check{u}(l,t) = \sum_{k=1}^{p} \alpha_k(t)\check{u}(\xi_k,t); \tag{51}$$

задача 6 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения $\check{u}(x,t)$ смешанной задачи для волнового уравнения (49) с финальными условиями (50) и с граничными условиями

$$\check{u}(0,t) = \mu(t),\ \check{u}_x(l,t) = \sum_{k=1}^{p} \alpha_k(t)\check{u}_x(\xi_k,t); \tag{52}$$

задача 7 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения $\check{u}(x,t)$ смешанной задачи для волнового уравнения (49) с финальными условиями (50) и с граничными условиями

$$\check{u}_x(0,t) = \mu(t),\ \check{u}(l,t) = \sum_{k=1}^{p} \alpha_k(t)\check{u}(\xi_k,t); \tag{53}$$

задача 8 об отыскании обобщенного из класса $\widehat{W}_2^2(Q_T)$ решения $\check{u}(x,t)$ смешанной задачи для волнового уравнения (49) с финальными условиями (50) и с граничными условиями

$$\check{u}_x(0,t) = \mu(t),\ \check{u}_x(l,t) = \sum_{k=1}^{p} \alpha_k(t)\check{u}_x(\xi_k,t); \tag{54}$$

Замечание. В задачах 5-8 финальные функции $\widehat{\varphi}(x)$ и $\widehat{\psi}(x)$ и граничная функция $\mu(t)$ должны удовлетворять следующим требованиям гладкости: $\widehat{\varphi}(x) \in W_2^2[0,l]$, $\widehat{\psi}(x) \in W_2^1[0,l]$, $\mu(t) \in W_2^2[0,T]$ в случае задач 5 и 6, $\mu(t) \in W_2^1[0,T]$ в случае задач 7 и 8. Кроме того, в случае задач 5 и 6 должно выполняться равенство $\widehat{\varphi}(0) = \mu(T)$.

Определение 7. Обобщенным из класса $\widehat{W}_2^2(Q_T)$ решением смешанных задач 5-8 называется такая функция $\check{u}(x,t)$ из класса $\widehat{W}_2^2(Q_T)$, которая удовлетворяет почти всюду волновому уравнению (49), финальным условиям (50) и соответствующим граничным условиям (51)-(54).

Замечание. Для всех четырех рассматриваемых смешанных задач 5-8 обобщенное решение $\check{u}(x,t)$ смешанной задачи с финальными условиями связано с обобщенным решением $u(x,t)$ соответствующей смешанной задачи с начальными условиями соотношением $u(x,t) = \check{u}(x,T-t)$.

Замечание. Отсюда и из доказанной основной теоремы следует, что *может существовать только одно обобщенное из класса* $\widehat{W}_2^2(Q_T)$ *решение* $\check{u}(x,t)$ *каждой из задач 5-8.*

Список литературы

[1] Ильин В. А. // Дифференц. уравнения. 2008. Т.44. №5. С.672-680.

[2] Ильин В. А., Моисеев Е. И. // Дифференц. уравнения. 2006. Т.42. №11. С.1558-1570.

[3] Ильин В. А., Моисеев Е. И. // Дифференц. уравнения. 2006. Т.42. №12. С.1699-1711.

[4] Ильин В. А., Моисеев Е. И. // Дифференц. уравнения. 2007. Т.43. №10. С.1369-1381.

[5] Ильин В. А., Моисеев Е. И. // Докл. РАН. 2005. Т.402. №2. С.163-169.

[6] Ильин В. А., Моисеев Е. И. // Докл. РАН. 2005. Т.402. №5. С.590-596.

[7] Ильин В. А., Моисеев Е. И. // Успехи матем. наук. 2005. Т.60, №6. С.89-114.

[8] Ильин В. А. // Дифференц. уравнения. 1999. Т.35. №11. С.1517-1534.

[9] Ильин В. А. // Дифференц. уравнения. 2000. Т.36. №11. С.1513-1528.

[10] Ильин В. А. // Дифференц. уравнения. 2000. Т.36. №12. С.1670-1686.

[11] Ильин В. А. // Успехи матем. наук. 1960. Т.15, №2. С.97-154.

[12] Никольский С. М. Приближение функций многих переменных и теоремы вложения. М.: Наука, 1969.

[13] Тихонов А. Н., Самарский А. А. Уравнения математической физики. М.: Наука, 2004.

[14] Владимиров В. С. Уравнения математической физики. М.: Наука, 1988.

УДК 681.322

БИБЛИОТЕЧНАЯ ПОДДЕРЖКА ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ МОДЕЛИ ЯЗЫКА РЕФАЛ

© 2009 г. И. Е. Бронштейн, А. В. Столяров

igor.bronstein@intelib.org, avst@cs.msu.ru

Кафедра алгоритмических языков

1 Особенности языка Рефал

Язык Рефал был предложен В. Ф. Турчиным в 1966 году [10]. В. Ш. Кауфман отмечает сходство изобразительных средств Рефала с нормальными алгоритмами Маркова [3]. Стилевую модель языка Рефал В. Ш. Кауфман называет *ситуационным программированием*, выделяя две ключевые абстракции — *анализ* исходной структуры данных и *синтез* результирующей структуры.

Рефал чрезвычайно удобен при решении задач, связанных с преобразованиями символьных данных. В. Ф. Турчин изначально позиционировал Рефал как *метаалгоритмический* язык [9] и позднее использовал его, в числе прочего, для изучения эквивалентных преобразований программ [11]. В. И. Сердобольский отметил удобство языка для работы с алгебраическими формулами [2]. С середины восьмидесятых годов и до настоящего времени Рефал успешно применяется в исследованиях, связанных с суперкомпиляцией и частичными вычислениями [12].

К настоящему времени существует несколько диалектов языка Рефал, среди которых наиболее динамично развиваются Рефал-5 [13] и Рефал Плюс. Разработаны методы эффективной реализации языка; созданы системы программирования для различных программно-аппаратных платформ.

Несмотря на это, Рефал крайне редко применяется в индустриальном программировании. При этом Рефал нельзя назвать неизвестным языком; многие профессиональные программисты с этим языком знакомы. Основной причиной слабой востребованности языка Рефал следует, видимо, считать ограниченность круга задач, в которых этот язык оказывается удобнее других; при этом для задач, не входящих в этот круг, применение Рефала хотя и возможно, но оказывается нецелесообразным.

С учётом этого становится актуальной задача *интеграции* вычислительной модели языка Рефал с системами программирования, основанными на других языках. В частности, можно рассмотреть один из самых популярных на сегодняшний момент индустриальных языков программирования — язык Си++ [8]; идея предоставления программистам в рамках проектов на Си++ выразительных возможностей, характерных для языка Рефал, выглядит достаточно привлекательной.

2 Библиотека InteLib

Библиотека InteLib (см., напр., [4, 6]), являясь библиотекой классов Си++, реализует *метод непосредственной интеграции*, впервые предложенный в работе [1]. Идея метода состоит в моделировании структур данных альтернативной вычислительной модели и введении для создаваемых классов такого набора операций, с помощью которого в тексте на базовом языке программирования (в данном случае Си++) можно формировать арифметические выражения, визуально напоминающие конструкции альтернативного языка. Реализовав соответствующий вычислитель, мы можем сделать такие выражения семантически эквивалентными исходным конструкциям.

В рамках библиотеки InteLib были реализованы модели языков Лисп и Scheme [7], делались также попытки создания моделей языков Плэнер [14] и Пролог. Созданная в 2001 году и описанная в статье [5] модель языка Рефал оказалась непригодна к практической работе в силу своей крайне низкой эффективности и не совсем удачно выбранного набора операций для формирования Рефал-выражений.

В настоящей статье описывается реализация вычислительной модели языка Рефал, имеющая более удобные средства задания Рефал-выражений и приемлемую (хотя и не очень высокую) эффективность по времени выполнения.

3 Представление конструкций Рефала выражениями Си++

Рефал является языком для работы со слабо структурированными данными. Единицами поля видимости здесь могут являться как символы (в традиционном смысле — как, например, символы в Си++), числа и «символы-метки», так и выражения в угловых или круглых скобках. Таким образом, уже на этапе выбора представления для этих конструкций в Си++ возникает вопрос, что должен хранить объект класса, представляющего единицу поля видимости, и как лучше организовать работу со скобочными структурами. Эти проблемы могут решаться двумя способами:

1. нелинейной структурой — скобочному выражению соответствует объект, содержащий указатель на структуру данных, представляющую выражение в скобках;

2. линейной структурой — как каждая из скобок, так и оставшиеся элементы выражения в скобках представляются отдельными объектами.

Существуют реализации языка Рефал, использующие каждый из этих способов. В рассматриваемой реализации отдано предпочтение линейным структурам, представленным в виде двусвязных списков. В этих списках вместе с объектами, соответствующими структурным и операторным скобкам, хранятся указатели на соответствующие им парные объекты; это позволяет ускорить навигацию по представлению выражения.

Список составляется из элементов структурного типа, названного `RfListItem`. В список, состоящий из таких элементов, могут входить объекты трёх основных типов: обычный (алфавитный) символ; элемент разметки (структурная или операторная скобка); все остальные объекты, а именно «символы-метки», числа, переменные и т. д., представляемые в виде произвольного S-выражения из ассортимента библиотеки InteLib.

Рефал-выражение как целое инкапсулируется в класс `RfExpression`. Объект этого класса хранит начало и конец двусвязного списка, представляющего выражение, и вводит операции для работы ним, такие как добавление новых элементов в конец списка, навигацию, извлечение элементов и т. п. Класс `RfExpression` входит в основную полиморфную иерархию библиотеки InteLib, его объекты заводятся в динамической памяти; в качестве основного интерфейса для работы с ними, как и с другими объектами этой иерархии, используются «умные указатели» [15], в данном случае — объекты класса `RfReference`.

Существования в InteLib класса `RfReference` уже достаточно, чтобы сформировать структуру данных, представляющую выражение Рефала, и затем обрабатывать её. Однако, как уже говорилось, целью метода непосредственной интеграции является реализация конструкций, которые синтаксически близки к исходным конструкциям интегрируемого языка. Для этого был введён статический класс `RfFormConstructor`. Конструкция «`1 2 3`», например, создаётся с помощью этого класса следующим образом:

```
RfFormConstructor R;
RfReference ref = (R|1, 2, 3);
```

Возможость использовать такую конструкцию обеспечивается переопределением операций. Например, для приведённого примера существенны две операции:

```
RfReference operator|(const SReference &r) const;
RfReference operator,(const SReference& ref);
```

Таблица 1: Представление конструкций Рефала выражениями Си++

Рефал-5	**Си++**
`1`	`1`
`'-'2`	`-2`
`'abcd'`	`"abcd"`
`symbol`	`symbol`
`"a symbol"`	`a_32_symbol`
`e.1`	`e_1`
`1 2 3 4`	`(R|1, 2, 3, 4)`
`(1 2 3 4)`	`(R|(R|1, 2, 3, 4))`
`<f 1 2 3 4>`	`(R<f, 1, 2, 3, 4>R)`
	`(~R)`
`()`	`(R|(~R))`

`f {`
`  1 = T;`
`  0 = F;`
`}`
	`f`
	`  || 1 ^ T`
	`  || 0 ^ F`

`f {`
`  e.1, <g e.1> :`
`  True = False`
`}`
	`f`
	`  || e_1 & (R|R<g, e_1>R) |`
	`  True ^ False`

Сначала операцией `operator|` (метод класса `RfFormConstructor`), применённой к `1`, создаётся объект класса `RfReference`, указывающий на список из одного элемента, который соответствует символу-числу `1`. Затем с помощью операции `operator,` (метод класса `RfReference`) к этому списку последовательно добавляются справа новые элементы.

Рассмотрим теперь представление функций языка Рефал. Тело функции состоит из набора предложений. Каждое такое предложение (если исключить из рассмотрения with-блоки, которые ранее не поддерживались в InteLib) можно записать в следующем виде:

```
pattern[1],
expression[1] : pattern[2],
...,
expression[N-1] : pattern[N] = expression[N],
```

Стоит, однако, заметить, что, заменив знак равенства на запятую и сгруппировав участвующие пары «образец — выражение», можно получить более единообразный вид, который позволит ввести удобную и простую структуру данных для хранения предложений и тела функций в целом:

```
pattern[1], expression[1] :
pattern[2], expression[2] :
... :
pattern[N], expression[N]
```

Для хранения таких пар в рассматриваемой реализации введён класс `RfClause`. Для представления предложения языка Рефал используется двусвязный список объектов этого класса. Для представления тела функции целиком потребовалось связать первые пары всех предложений также в односвязный список: в `RfClause` был введён атрибут, указывающий для первой пары текущего предложения на первую пару следующего, либо равный `0` в остальных случаях.

В случае с функциями языка Рефал для обеспечения синтаксической схожести моделирующих и исходных конструкций также используется переопределение арифметических операций. Идея состоит в том, чтобы для объекта, представляющего символ Рефала, иметь возможность в начале выполнения программы указать, какая точно конструкция, представляющая тело функции, ему соответствует. Например, пусть дана функция:

Рефал:

```
Pal {
  s.1 e.2 s.1 =
    <Pal e.2>;
  = True;
  s.1 = True;
  e.1 = False;
}
```

Си++:

```
Pal
    || (R|s_1, e_2, s_1) ^
       (R|R<Pal, e_2>R)
    || (~R) ^ (R|True)
    || (R|s_1) ^ (R|True)
    || (R|e_1) ^ (R|False)
```

Рис. 1: соответствие конструкций, описывающих тело функции на Рефале-5 и на Си++

```
f1 {
  s.1 s.2, <f2 s.1> : True = True;
  e.1 = False
}
```

Ей будет соответствовать следующий текст программы в конструкциях библиотеки InteLib:

```
RfSymbol f1("f1");
RfSymbol f2("f2");
f1 || (R|s_1, s_2) & (R<f2, s_1>R) | (R|True) ^ (R|True)
   || (R|e_1) ^ (R|False);
```

Для облегчения понимания того, как после применения переопределённых арифметических операций объект класса `RfSymbol` становится связанным со структурой, описанной выше, две последние строчки следует переписать с учётом приоритета операций в Си++:

```
RfSymbol f1("f1");
RfSymbol f2("f2");
f1 || (((R|s_1, s_2) & (R<f2, s_1>R)) | ((R|True) ^ (R|True))))
   || ((R|e_1) ^ (R|False);
```

Существенными для приведённого примера являются следующие переопределённые операции:

```
RfClause operator&(RfReference &pattern, RfReference &result);
RfClause operator^(RfReference &pattern, RfReference &result);
RfClause& operator|(const RfClause& a);
RfClause& operator||(const RfClause &cla);
```

Сначала с помощью операций `operator&` и `operator^` (которые выполняют одни и те же действия) создаётся новый объект класса `RfClause`, в котором «склеены» в пару образец и выражение; затем с помощью оперции `operator|` образуются односвязные списки, представляющие предложение; наконец, на третьем этапе выполняются операции `operator||`. Первая из них связывает символ `f1` с первым предложением-списком, остальные последовательно «присоединяют» первую пару нового предложения к уже существующей конструкции.

Такие операторы инициализации всех символов, соответствующих функциям в Рефале, удобно поместить в одну процедуру, которая будет вызываться в начале программы перед непосредственным вызовом этих функций.

4 Внутренние структуры данных

Опишем основные структуры данных рефал-вычислителя.

Связи между переменными и соответствующими значениями хранятся в переменных структурного типа `RfBinding`, в котором предусмотрены указатели на связываемую переменную и элементы выражения, в котором находится соответствующее значение.

Ключевым понятием при реализации рефал-машины является понятие диапазона. В диапазоне указываются начало и конец разбираемых в каждый конкретный момент участков образца и выражения. При этом там также могут содержаться ссылки на уже просмотренные диапазоны либо на диапазоны, которые только предстоит просмотреть. Отдельной задачей является выбор оптимальной структуры данных для хранения диапазонов. В рассматриваемой реализации в качестве такой структуры выбрано двоичное дерево — «дерево диапазонов».

Корнем дерева является главный диапазон — диапазон, соответствующий целым образцу и выражению. Во всех узлах помимо соответствующих указателей на образец и выражение хранятся указатели на левого и правого потомка, а также на родителя данного узла; соответствующая структура называется `RfRange`.

Все объекты классов, соответствующих абстракциям, используемым при реализации рефал-вычислителя (связи между переменными и их значениями, сохранённые выражения, диапазоны сопоставления и другие), объединены в общей структуре данных — контексте, который реализуется классом `RfContext`.

5 Реализация Рефал-вычислителя

В рамках Рефал-вычислителя нам необходимо реализовать сопоставление образца и выражения, подстановку значений переменных в заданное выражение, обработку отдельного предложения Рефал-программы и, наконец, шаг работы Рефал-машины, представляющий собой замену в поле зрения вызова функции на результат такого вызова.

5.1 Сопоставление

Сигнатура функции сопоставления выражений выглядит следующим образом:

```
bool MatchProcess(RfContext &context, RfRange *main_range);
```

Аргумент `context` представляет собой контекст, который будет изменяться в ходе выполнения функции сопоставления, а `main_range` — диапазон, с которого нужно начинать обход дерева.

Дерево диапазонов обходится в глубину («левый потомок — узел — правый потомок»). При этом обход считается произведённым успешно, если успех достигнут в каждом узле дерева. Если сопоставление в каком-либо узле оказалось неуспешным, неуспех выдается всей функцией сопоставления. Однако это не значит, что всё дерево диапазонов уничтожается. Оно сохраняется в списке деревьев (в контексте), и в дальнейшем к нему может быть осуществлён возврат. Сопоставление в каждом конкретном узле считается успешным, когда текущие участки и образца, и выражения пусты.

В каждый конкретный момент выполнения функции сопоставления специально выделен один из узлов дерева диапазонов, называемый рабочим диапазоном. Однако ни он, ни другие узлы не меняются в процессе выполнения функции сопоставления. Все модификации производятся в текущем диапазоне — объекте, в который копируются все данные из только что выбранного рабочего диапазона.

Если в образце встретился символ, алгоритм проверяет наличие такого же символа в соответствующем месте выражения. Если выражение в круглых скобках, то проверяется, есть ли соответствующая скобочная структура в выражении. Если она отсутствует, то сопоставление считается неуспешным. Иначе вносятся изменения в дерево диапазонов: к рабочему узлу добавляются два потомка. Они соответствуют двум половинам образца и, соответственно, выражения, на которые их «разбивает» скобочная структура. Далее сопоставление в узле дерева считается успешным, и новым рабочим диапазоном становится левый потомок старого.

Если в образце встретилась уже связанная с некоторым значением переменная, алгоритм сопоставления сводится к попытке найти соответствующее значение в выражении, соответствующим образом изменить текущий диапазон (если значение найдено) и продолжить работу с ним. Если же ещё не связанная с некоторым значением переменная типа `s` или `t`, то она связывается с соответствующим символом или термом из выражения, если это возможно.

Соответствущая связь «переменная — значение» добавляется в начало списка связей. Далее функция сопоставления продолжает работу с модифицированным текущим диапазоном.

Переменные типа `e`, ещё не связанные с каким-либо значением, могут быть либо закрытыми, либо открытыми. Если в текущем диапазоне в образце осталась лишь одна не связанная e-переменная, то ей сразу можно сопоставить всё соответствующее диапазону выражение. Такая `e`-переменная называется закрытой. В противном случае она является открытой. Алгоритм сопоставления старается избегать открытых переменных. Если такая переменная встретилась при сопоставлении слева направо, направление сопоставления меняется на противоположное и попытка повторяется. Таким образом, необходимость работы с новой переменной типа `e` возникает лишь тогда, когда в текущем участке образца переменных этого типа больше одной и они находятся как в начале, так и в конце участка.

Если в образце встретилась открытая переменная типа `e`, то она исходно считается связанной с пустым значением. При этом в целях обеспечения возможности поиска с возвратом для этой переменной хранятся указатели на конец текущего хранимого значения (`0` в случае пустого значения) и на конец теоретически максимально возможного значения. В дальнейшем в случае поиска с возвратом первый из этих указателей будет при необходимости пробегать все позиции от начала до конца сопоставляемого выражения. Соответствущая связь «переменная — значение» добавляется в начало списка связей.

В случае открытой переменной помимо добавления новой связи так же, как и при сопоставлении скобочных структур, меняется дерево диапазонов. У рабочего узла дерева появляется один потомок, в который копируются данные текущего диапазона. Кроме того, создаются перекрёстные ссылки между только что добавленным диапазоном и связью «переменная — значение». Новый потомок становится рабочим диапазоном, и функция сопоставления продолжает свою работу.

Всё описанное ранее относилось к случаю, когда функция сопоставления начинает создавать дерево диапазонов с самого начала, с главного диапазона. Однако также существует необходимость вызывать эту функцию в случае возникновения поиска с возвратом. В этом случае дерево диапазонов уже построено, но требуется изменение значений некоторых переменных, что может, в свою очередь, привести к модификации дерева.

В первую очередь выбирается переменная, значение которой можно изменить для осуществления поиска с возвратом. Из выбора структур данных для хранения связей «переменная — значение» следует, что такой переменной (если она существует) является первая в списке связей переменная типа `e`, для которой номер фрейма совпадает с текущим номером (т. е. получившая значение в текущем фрейме), а указатель текущего значения не достиг последнего возможного значения. Далее значение это переменной изменяется на следующее возможное, и функции сопоставления в качестве `main_range` передаётся не корень существующего дерева диапазонов, а диапазон, соответствующий выбранной переменной. Связи, находящиеся в списке до выбранной переменной, удаляются, так как значения соответствующих переменных могут измениться. Таким образом, необходимо модифицировать дерево диапазонов с тем, чтобы ещё раз осуществить попытку сопоставления с новым значением выбранной переменной. Из свойств дерева диапазонов следует, что для этого необходимо лишь вновь обойти дерево в глубину, начиная с рабочего диапазона.

Необходимо отметить существование различных возможностей для оптимизации приведённой выше реализации. Например, в ней для переменных типа `e` могут перебираться все варианты значений, в том числе и заведомо невозможные. Так, если справа от этой переменной в образце находится (сразу или после нескольких открывающихся круглых скобок) некоторый символ, называемый подсказкой, перебор значений возможно производить гораздо более оптимально. Достаточно хранить подсказку и соответствующее число скобок до неё и при каждом переходе к новому значению непосредственно проверять, возможно ли оно. Это называется оптимизацией с помощью подсказки.

5.2 Подстановка

Сигнатура функции, осуществляющей преобразование рефальского выражения с учётом подстановки в него значений переменных, выглядит следующим образом:

```
RfReference RefalSubstitute(const RfReference &right_part,
                            RfContext &context,
                            bool is_last);
```

Здесь `right_part` — это выражение, в котором необходимо произвести замену, `context` — текущий контекст, а смысл `is_last` станет понятен несколько позже. Функция возвращает выражение, в котором уже произведены соответствующие замены.

Простейшая реализация этой функции, осуществлённая изначально, выглядит следующим образом. Вначале создаётся новое выражение языка Рефал, в которое затем будут добавляться элементы. Затем осуществляется проход по всем объектам класса `RfListItem`, составляющим выражение. Здесь возможны два случая: если встретился объект, не являющийся переменной, необходимо создать его копию; иначе необходимо скопировать значение встретившейся переменной.

Однако, такая реализация является неэффективной. Рассмотрим, например, некоторую функцию, которая разбирает выражение посимвольно и осуществляет некоторую операцию с каждым символом (подобные функции очень типичны для лексического анализа текста и т. д.):

```
SomeFunction {
    s.1 e.2 = <MakeSomethingWith s.1> <SomeFunction e.2>;
    = ;
}
```

Фактически для вышеописанной реализации в приведённой функции необходимо было бы совершить $N - 1 = O(N)$ (где N — длина поля видимости) дорогих операций копирования объектов класса `RfListItem`. Всего же за время выполнения функции потребовалось бы $1 + 2 + \cdots + (N - 1) = O(N^2)$ операций копирования.

В связи с этим возникла необходимость в оптимизации функции замены. Для этого был добавлен новый аргумент `is_last`, который равен `true` в случае, если функция замены гарантированно вызывается последний раз в текущем предложении (т. е. в том случае, если выполняется замена после знака равенства, а не в where-предложении после запятой). Кроме того, для каждой связи «переменная — значение» была добавлена булевская переменная `was_used`, изначально (при добавлении связи) равная `false`.

Далее каждый раз, когда возникает необходимость копировать значение переменной типа `e` либо переменной типа `t` (в том случае, если её значением является не символ, а терм), в первую очередь проверяется, нельзя ли использовать это значение в правой части непосредственно, без копирования. Для каждой переменной такая возможность существует лишь один раз, поэтому осуществляется проверка `was_used` на `false`. Если `was_used` равно `true`, то осуществляется обычное копирование, описанное выше. Иначе `was_used` присваивается `true`, и выполняется следующее: участок, соответствующий значению переменной, «вырезается» из изначального списка и добавляется в конец генерируемого (т. е. происходит использование значения без копирования).

Оптимизированная реализация является гораздо более эффективной. В общем случае, если в правой части предложения нет повторяющихся переменных, число операций копирования теперь не зависит от размера поля видимости и ограничено константой — общим числом переменных в правой части. Итак, для приведённой выше функции на каждом шаге будет выполняться всего одно копирование. Таким образом, в ходе выполнения всей функции потребуется лишь N, а не $O(N^2)$ (где N — длина поля видимости) подобных операций копирования.

5.3 Реализация функциональных вызовов

Сигнатура функции, которая реализует этап работы рефал-вычислителя, связанный с одним конкретным предложением, выглядит следующим образом:

```
bool RfClause::EvaluateClause(RfReference &view,
                              RfReference &res,
                              RfContext &context) const;
```

Данная функция является методом класса `RfClause`. Аргумент `view` представляет собой рефальское выражение, к которому будет применяться соответствующее объекту `RfClause` предложение. Аргумент `context` — это контекст, смысл передачи которого в функцию `EvaluateClause` станет понятен при описании реализации with-блоков (безымянных функций). Результирующее выражение записывается в `res`.

Сопоставление начинается с первой пары, предствляющей предложение, и выбирается направление движения слева направо. В результате работы функции либо достигается успех в последнем узле списка — в этом случае успешным считается всё предложение, либо фиксируется неуспех в первом узле — при отсутствии каких-либо дальнейших возможностей отката назад. Соответствующий результат (успех или неуспех) возвращается функцией `EvaluateClause`.

В одной из локальных переменных хранится текущее сопоставляемое выражение, которое исходно равно выражению, передаваемому в функцию через параметр `view`.

Если в какой-то момент просматривается узел и направлением обхода является направление слева направо, то производится попытка сопоставления текущего сопоставляемого выражения и образца из текущей пары.

Если же направлением обхода является направление справа налево, это значит, что в этот узел был произведён возврат из последующих узлов из-за неудачного сопоставления в них. В этом случае производится описанная в § 5.1 процедура поиска нужной переменной, значение которой возможно изменить.

Если такая переменная найдена, то направление обхода меняется на направление слева направо. Если же нет — необходимо откатиться к предыдущей в списке паре, если таковая существует. В случае, когда пара была первой в списке, фиксируется неуспех всего предложения.

Далее приведена сигнатура функции, реализующей вычисление не одного предложения, а рефальской функции целиком:

```
void RfClause::Apply(RfReference &view, RfContext &context) const;
```

Здесь `view` — это выражение, к которому применяется функция. Результат выполнения функции также записывается в `view`.

В этой функции последовательно применяются `EvaluateClause` для первых пар списков, представляющих рефальские предложения, и результатом работы является результат работы первой завершившейся успехом функции, связанной с конкретным предложением. Если такое предложение не было найдено, возбуждается исключительная ситуация.

Отдельно необходимо сказать о реализации в библиотеке InteLib with-блоков. Фактически они представляют собой безымянные функции, в которые передаётся в качестве аргумента последнее вычисленное в предложении выражение. Следует отметить, что в отличие от обычных функций, перед вызовом которых создаётся новый контекст, при вызове безымянных функций контекст используется существующий. Значения, с которыми были связаны переменные во внешней функции, не могут быть изменены во внутренней.

Для поддержки with-блоков был создан новый статический класс `RfWith`, инкапсулирующий в себе список безымянных символов и соответствующих им функций.

Рассмотрим для примера функцию на языке Рефал, содержащую with-блок:

```
Go {
  s.1 s.2, <f s.1> : { True = s.1; False = s.2 }
}
```

Ниже представлено выбранное для неё представление в виде выражения Си++:

```
RfWith RWITH;

Go
  || (R|s_1, s_2) & (R|R<f, s_1>R) |
  (RWITH
```

```
    || (R|True)        ^ (R|s_1)
    || (R|False)       ^ (R|s_2)
  );
```

В объекте класса `RfWith` хранится список безымянных символов. Операция `operator||` генерирует новый безымянный символ и добавляет его в начало списка.

Для поддержки with-блоков в класс `RfClause` был добавлен булевский атрибут `is_where`, значение которого равно `true`, если пара ссылается на безымянную функцию, и `false` в противном случае.

При вызове операции `operator|` создаётся пара «образец — выражение» следующего вида: значение `is_where` в ней равно `true`, а в `pattern` хранится непосредственно ссылка на соответствующую безымянную функцию.

Когда текущей парой становится пара такого специального вида, вызывается соответствующая безымянная функция от текущего сопоставляемого выражения. В эту функцию передаётся не вновь созданный (как в случае обычных функций), а уже существующий контекст. Перед этим происходит операция обновления контекста. Она заключается в том, что во всех связях между переменными и значениями номеру фрейма связи присваивается значение `0`. Это гарантирует, что значения таких переменных никогда не изменятся во вложенном блоке, что удовлетворяет семантике блоков.

Благодаря реализации with-блоков и функций стандартной библиотеки было достигнуто соответствие поддерживаемого диалекта описанию языка Рефал-5.

6 Транслятор IRINA

Транслятор IRINA (**I**nteLib **R**efal **I**ntelligent, **N**ew and **A**ll-inclusive Translator) принимает на вход один или несколько файлов на Рефале. На выходе генерируется модуль языка Си++ в виде заголовочного файла и файла реализации.

Для настройки параметров IRINA используются директивы трансляции. Синтаксис каждой директивы следующий:

```
НАЗВАНИЕ_ДИРЕКТИВЫ = аргумент
```

либо

```
НАЗВАНИЕ_ДИРЕКТИВЫ аргумент1 = аргумент2
```

Список директив представляет собой последовательность таких псевдопредложений, ограниченную конструкцией `/*%%%...*/`:

```
/*%%%
    ДИРЕКТИВА[1];
    ...
    ДИРЕКТИВА[N];
*/
```

Необходимо отметить важное свойство списков директив такого вида: они являются прозрачными для стандартного транслятора Рефала-5, так как с его точки зрения представляют собой обычный многострочный комментарий. Это позволяет избегать модификации файлов при работе как со стандартной системой программирования на Рефале-5, так и с транслятором IRINA.

Конструкции `/*%%%...*/` могут встречаться в одном файле более одного раза: при этом списки директив, заключённые внутри них, просто считаются объединёнными в общий список.

Директивы трансляции позволяют, в частности: объявлять публичные и/или внешние символы или функции для текущего модуля на Рефале; управлять правилами преобразования имён переменных и символов; задавать имена результирующих файлов; задавать дополнительные настройки, связанные с содержимым результирующего файла реализациии (`.cxx`).

Основная часть транслятора IRINA состоит из нескольких модулей на Рефале, каждый из которых реализует свой этап трансляции: препроцессирование, синтаксический и семантический анализ, редактирование межмодульных связей и, наконец, генерацию выходных файлов.

7 Заключение

Описываемая реализация всё ещё уступает по времени исполнения оригинальным реализациям Рефала, что вполне объяснимо с учётом используемого в ней интерпретируемого характера выполнения рефал-программ, тогда как большинство реализаций Рефала компилируемые. В ходе проводившегося тестирования рассматриваемая реализация работала медленнее, чем реализация Рефала-5, в 8–12 раз. Следует отметить, однако, что предыдущая InteLib-реализация Рефала проигрывала Рефалу-5 в 90–100 раз, так что достигнутые результаты можно рассматривать как обнадёживающие. Оставшийся проигрыш может быть частично компенсирован возможностью реализации части подзадач на базовом языке проекта (на Си++).

Рефал-подсистема в описанном варианте включена в библиотеку InteLib, начиная с версии 0.6.20.

Список литературы

[1] Bolshakova E. Stolyarov A. Building functional techniques into an object-oriented system // Knowledge-Based Software Engineering. Proceedings of the 4th JCKBSE, vol. 62 of Frontiers in Artificial Intelligence and Applications, pages 101–106, Brno, Czech Republic, September 2000. IOS Press, Amsterdam, 2000.

[2] Сердобольский В. И. Язык РЕФАЛ и его использование для преобразования алгебраических выражений // Кибернетика. 1969. №3, стр. 45–51.

[3] Кауфман В. Ш. Языки программирования. Концепции и принципы. М., Радио и связь, 1993.

[4] Столяров А. В. Интеграция изобразительных средств альтернативных языков программирования в проекты на C++. Рукопись депонирована в ВИНИТИ РАН 06.11.2001, № 2319-В2001, Москва, 2001.

[5] Столяров А. В. Расширенный функциональный аналог языка Рефал для мультипарадигмального программирования // Л. Н. Королёв, ред., Программные системы и инструменты. Тематический сборник, выпуск 2, стр. 184–195. Издательский отдел факультета ВМиК МГУ, Москва, 2001.

[6] Столяров А. В. Библиотека InteLib — инструмент мультипарадигмального программирования. Тезисы докладов II конференции разработчиков свободных программ «На Протве», Обнинск, 25–27 июля 2005 года, стр. 56–62.

[7] Столяров А. В. Импорт вычислительной модели языка Scheme в объектно-ориентированное окружение // Сборник статей молодых учёных факультета ВМиК МГУ, № 5. М.: Издательский отдел факультета ВМиК МГУ, 2008, стр. 119–130.

[8] Stroustrup B. The Design and Evolution of C++. Addison-Wesley, 1994. 462 pages.

[9] Турчин В. Ф. Метаалгоритмический язык. // Кибернетика. 1968. N 4. Стр. 45-54.

[10] Турчин В. Ф. Программирование на языке Рефал. М.: ИПМ АН СССР, 1971.

[11] Турчин В. Ф. Эквивалентные преобразования программ на Рефале // Автоматизированная система управления строительством. М.: ЦНИПИАСС, 1974, стр. 36–68.

[12] Turchin V. F. The concept of a supercompiler // ACM Transactions on Programming Languages and Systems. 1986. Vol. 8, N 3. Pg. 292-325.

[13] Turchin V. F. Refal-5, Programming Guide and Reference Manual. New England Publishing Co., Holyoke, 1989.

[14] Фролова О. Г. Библиотечная реализация вычислительной модели языка Плэнер // Сборник статей молодых учёных факультета ВМиК МГУ, № 5. М.: Издательский отдел факультета ВМиК МГУ, 2008, стр. 24–33.

[15] Элджер Дж. C++: библиотека программиста. СПб.: Питер, 2002.

УДК 519.7

ФУНКЦИЯ ШЕННОНА ДЛИНЫ ПРОВЕРЯЮЩИХ ТЕСТОВ ФУНКЦИЙ, БЕСПОВТОРНЫХ В ЭЛЕМЕНТАРНОМ БАЗИСЕ

© 2009 г. С. Е. Бубнов

sergey.msu@gmail.com

Кафедра Математической Кибернетики

1 Введение. Постановка задачи

В работе исследуется длина проверяющих тестов для функций, бесповторных в базисе $\{0, 1, \&, \vee, \neg\}$. Задача построения тестов как логический метод контроля управляющих систем описана в [1]. По всей видимости, самым ранним упоминанием задачи содержится в работе тех же авторов [2].

Определения. Будем говорить, что *две функции от одинаковых переменных отличаются на множестве T наборов этих переменных или, короче, отличаются на множестве наборов T*, тогда и только тогда, когда существует набор из T, на котором функции принимают разные значения. *Проверяющий* тест для функции – множество наборов, достаточное для того, чтобы отличить на нем заданную функцию от некоторого заранее фиксированного множества функций. *Длиной* теста называется количество наборов в нем. *Минимальным* будем называть проверяющий тест, имеющий наименьшую длину среди всех проверяющих тестов функции.

Пусть задан базис B. Функция называется *бесповторной* в B, если она выразима бесповторной формулой (то есть такой, в которой символ каждой переменной встречается не более одного раза) в B. Описание свойств бесповторных булевых функций можно найти, например, в монографии [3]. Часто бесповторными называются функции, выразимые бесповторными формулами в базисе всех функций, существенно зависящих не более чем от двух переменных (далее такой базис обозначается через B_2). Исследованию длины проверяющих тестов бесповторных в B_2 функций посвящена работа [4]. Однако, поскольку в дальнейшем рассматриваются функции, бесповторные в базисе $B_0 = \{0, 1, \&, \vee, \neg\}$, то для удобства изложения, бесповторными здесь будут называться именно такие функции. Базис B_0 также принято называть *элементарным* (см. [5]). *Тестом относительно бесповторной альтернативы* (далее просто *тестом*) для бесповторной функции $f(x_1, \ldots x_n)$, существенно зависящей от всех своих переменных, будем называть проверяющий тест для f на множестве всех бесповторных функций, зависящих от n переменных произвольным образом. Длину минимального теста относительно бесповторной альтернативы для функции f будем обозначать через $T_{B_0}(f)$. Через $T_{B_0}(n)$ будем обозначать функцию Шеннона:

$$T_{B_0}(n) = \max_{f(x_1, \ldots x_n)} T_{B_0}(f).$$

Впервые сложность тестирования бесповторных в элементарном базисе функций была исследована в статье [6], где были получены линейные оценки для $T_{B_0}(n)$. А именно, доказывается

Утверждение 1. *(см. [6])*

$$n + 1 \leqslant T_{B_0}(n) < 3.5n.$$

В настоящей работе доказывается

Теорема 1.

$$T_{B_0}(n) = n + 1.$$

2 Основная часть

Бесповторные функции можно представить в виде корневого дерева, каждая внутренняя вершина которого помечена одним из символов & или $\vee$, а листья помечены символами переменных или их отрицаний. При этом смежные вершины не могут быть помечены одинаково. Такие деревья называются *каноническими* (см. [4], [6]). Функция, сопоставляемая дереву, определяется от листьев к корню по следующему правилу: функция, реализуемая в некоторой вершине, получается в результате применения соответствующей пометке операции от переменных – функций, реализуемых в нижележащих вершинах (будем располагать корень дерева над листьями при описании канонических деревьев). Функции-константы представляются в виде деревьев, состоящих ровно из одной вершины, помеченной символом этой константы. Канонические деревья полезны тем, что они взаимнооднозначно сопоставляются своим бесповторным функциям (см. [4]).

Введем теперь понятие *наследственно-несвязного* графа (*НН-графа*).

Определение. Граф, состоящий ровно из одной вершины является наследственно-несвязным. Пусть в графе более одной вершины, тогда он является наследственно-несвязным, если и только если либо он сам, либо его дополнение несвязны и компоненты связности из которых состоит либо он, либо его дополнение являются также наследственно-несвязными графами.

Утверждение 2. *Любой подграф НН-графа также является НН-графом.*

Доказательство. Проведем доказательство индукцией по количеству вершин в графе. Для графа, состоящего из двух вершин утверждение, очевидно, выполнено. Пусть теперь имеем НН-граф с числом вершин $k > 2$ и утверждение выполняется для всех связно-несвязных графов с числом вершин меньше k. Тогда достаточно показать свойство наследственной несвязности любого подграфа с числом вершин $k-1$, что соответствует удалению некоторой вершины с инцидентными ей ребрами из исходного графа.

Рассмотрим дополнение графа, если оно несвязно, иначе перейдем к рассмотрению самого графа. Пусть удаляемая вершина принадлежит компоненте связности, состоящей из более чем одной вершины, тогда, согласно предположению индукции, удалением этой вершины получим также НН-граф, что показывает, что все компоненты связности графа (или его дополнения) являются НН-графами, а следовательно и получившийся граф является НН-графом. Если же компонента связности состоит ровно из одной вершины, то эта вершина является изолированной и ее удаление сохранит граф наследственно-несвязным. Утверждение доказано.

Утверждение 3. *(Правило пирамиды, см. [7])*
Если дополнение четырехвершинного графа G, имеющего цепочку из трех ребер, несвязно и граф G не содержит два из оставшихся ребер, то G содержит третье ребро.

Доказательство. Для доказательства этого утверждения достаточно заметить, что любая цепочка из трех ребер в четырехвершинном графе связывает все его вершины, а полный четырехвершинный граф можно разбить на две такие цепочки. Утверждение доказано.

Указанное правило позволяет восстанавливать некоторую информацию о НН-графе по неполной информации о его ребрах. Рассмотрим граф, получаемый из канонического дерева по следующему правилу: вершинам графа сопоставляются листовые вершины дерева, причем вершины графа связаны ребром тогда и только тогда, когда ближайшая на пути из соответствующих листовых вершин в корень общая вершина помечена символом &. Нетрудно показать, что для канонических деревьев, соответствующих функциям, отличным от констант, граф, построенный по такому алгоритму будет являться наследственно-несвязным. При этом по графу можно восстановить однозначно само каноническое дерево с точностью до пометок листовых вершин. Будем обозначать этот граф $G_{\&}$ (см. рис. 6 на стр. 54).

Без потери общности рассуждений далее всюду будем считать, что функция $f = f(x_1, \ldots, x_n)$, для которой мы будем строить тест, является монотонной по всем переменным, т.е. в каноническом дереве отсутствуют листья, помеченные отрицанием какой-либо переменной. Возможность такого допущения следует из того, что $T_{B_0}(f(x_1, \ldots, x_n)) = T_{B_0}(f(x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}))$, для произвольного набора степеней $\sigma = (\sigma_1, \ldots, \sigma_n)$. Перейдем к основной части доказательства теоремы 1.

Будем считать деревья, состоящие ровно из одной вершины, деревьями высоты 1.

Построим для произвольного канонического дерева D бесповторной монотонной функции n переменных множество наборов $S(D)$, поэтапно строя для каждого его поддерева d по два набора размерности n: $\tilde{o}(d)$ и $\tilde{e}(d)$.

1) Висячей вершине, которая помечена символом переменной x_i, положим $\tilde{o} = (x_i = 0)$, $\tilde{e} = (x_i = 1)$.

2) Пусть уже построили пары наборов $\tilde{o}'(d)$ и $\tilde{e}'(d)$ для всех деревьев высоты не больше h, определив значения только на тех переменных, от которых существенно зависит функция, реализуемая в корне d. Пусть у дерева D высоты $h+1$ имеется r поддеревьев $d_1, \ldots, d_r$, соединенных с корнем ребрами. Если корень D помечен символом & ($\vee$), тогда значения переменных в $\tilde{e}(D)$ ($\tilde{o}(D)$) положим равными значениям соответствующих переменных в наборах $\tilde{e}'(d_1), \ldots, \tilde{e}'(d_r)$ ($\tilde{o}'(d_1), \ldots, \tilde{o}'(d_r)$):

$$\tilde{e}(D) = (\overline{\tilde{e}`(d_1), \ldots, \tilde{e}`(d_r)})$$

$$\left(\tilde{o}(D) = (\overline{\tilde{o}'(d_1), \ldots, \tilde{o}'(d_r)})\right)$$

Здесь чертой над набором обозначена операция конкатенации наборов. К примеру, результатом применения операции конкатенации к наборам различных переменных $a = (a_1, b_1)$ и $b = (b_1, b_2)$ в зависимости от порядка операндов будет:

$$\overline{ab} = (a_1, a_2, b_1, b_2), \text{ или}$$

$$\overline{ba} = (b_1, b_2, a_1, a_2).$$

Чтобы получить из наборов $\tilde{e}'(d_1), \ldots, \tilde{e}'(d_r)$ ($\tilde{o}'(d_1), \ldots, \tilde{o}'(d_r)$) наборы $\tilde{e}(d_1), \ldots, \tilde{e}(d_r)$ ($\tilde{o}(d_1), \ldots, \tilde{o}(d_r)$), увеличим их размерность до n и доопределим значениями переменных из $\tilde{e}(D)$ ($\tilde{o}(D)$). Заметим, что получившиеся наборы равны между собой.

Построим теперь наборы $\tilde{o}(d_1), \ldots, \tilde{o}(d_r)$ ($\tilde{e}(d_1), \ldots, \tilde{e}(d_r)$). Если есть поддерево d_j высоты 1, то в качестве $\tilde{o}(d_j)$ ($\tilde{e}(d_j)$) берем следующий набор:

$$\tilde{o}(d_j) = (\overline{1, \ldots, 1, \tilde{o}'(d_j), 1 \ldots, 1})$$

$$\left(\tilde{e}(d_j) = (\overline{0, \ldots, 0, \tilde{e}'(d_j), 0, \ldots, 0})\right)$$

Для произвольного дерева d_i, у которого высота больше 1, используем вместо единиц (нулей) значения переменных из набора $\tilde{e}(D)$ ($\tilde{o}(D)$):

$$\tilde{o}(d_i) = (\overline{\tilde{e}'(d_1), \ldots, \tilde{e}'(d_{i-1}), \tilde{o}'(d_i), \tilde{e}'(d_{i+1}) \ldots, \tilde{e}`(d_r)})$$

$$\left(\tilde{e}(d_i) = (\overline{\tilde{o}'(d_1), \ldots, \tilde{o}'(d_{i-1}), \tilde{e}'(d_i), \tilde{o}'(d_{i+1}) \ldots, \tilde{o}'(d_r)})\right)$$

Если существуют деревья из $d_1, \ldots, d_r$, у которых высота больше 1, то в качестве $\tilde{o}(D)$ ($\tilde{e}(D)$) берем любой из получившихся наборов $\tilde{o}(d_i)$. В случае, если все поддеревья – висячие вершины, то в качестве набора $\tilde{o}(D)$ ($\tilde{e}(D)$) берем любой из наборов $\tilde{o}(d_1), \ldots, \tilde{o}(d_r)$ ($\tilde{e}(d_1), \ldots, \tilde{e}(d_r)$).

Итак, построили наборы размерности n $\tilde{o}(D)$ и $\tilde{e}(D)$, $\tilde{o}(d_1), \ldots, \tilde{o}(d_r)$ и $\tilde{e}(d_1), \ldots, \tilde{e}(d_r)$. Чтобы получить наборы $\tilde{o}(d)$ и $\tilde{e}(d)$ размерности n для любого поддерева d, не входящего во множество $\{D, d_1, \ldots, d_r\}$, доопределяем $\tilde{o}'(d)$ и $\tilde{e}'(d)$, используя значения переменных из набора $\tilde{e}(D)$. Результирующим множеством будет являться объединение всех наборов $\tilde{o}$ и $\tilde{e}$ размерности n всех поддеревьев исходного канонического дерева. Построение окончено.

На рисунках 1–4 приведен пример построения множества наборов $S(D)$ для всех поддеревьев канонического дерева для функции $f = x_1x_2(x_3 \vee x_4)(x_5 \vee x_6x_7)$. Сначала строятся наборы для поддеревьев высоты один (рис. 1), представляющих из себя вершины помеченные переменными, потом – два (рис. 2), и так далее.

Рис. 1: функции, реализуемые в деревьях высотой 1: все переменные

Рис. 2: функции, реализуемые в деревьях высотой 2: f_2,f_4

Рис. 3: функции, реализуемые в деревьях высотой 3: f_3

Рис. 4: функции, реализуемые в деревьях высотой 4: f_1

В получившемся множестве $S(D)$ число различных наборов равно восьми, что совпадает с нижней границей $T_{B_0}(7)$ (см. утв. 1).

Ввиду неоднозначности отображения $S(D)$, в дальнейшем, если не оговорено иное, будем полагать, что на каждом шаге построения $S(D)$ при имеющемся выборе между несколькими наборами брался лексикографически наименьший набор. Аналогично можно определить отображение $S(f)$ для не являющимися константами функций, так как для каждой бесповторной функции существует единственное каноническое дерево.

Заметим, что на наборах $\tilde{e}$ функция принимает значение 1, а на $\tilde{o}$ – 0. Помимо этого выполняется

Утверждение 4. *Для любого поддерева d высоты больше 1 наборы $\tilde{o}(d)$ и $\tilde{e}(d)$ отличаются в значении ровно одной переменной.*

Доказательство. Докажем по индукции высоты поддеревьев.

Для деревьев высоты 2 утверждение проверяется непосредственно.

Пусть верно для всех канонических деревьев высоты не более $h \geqslant 2$, докажем для произвольного канонического дерева высоты $h+1$. Пусть для простоты корень помечен символом & и в дереве r поддеревьев $d_1, \ldots, d_r$ соединены с корнем ребрами. По условию, одно из поддеревьев имеет высоту больше, чем один. В этом случае наборы $\tilde{o}(D)$ и $\tilde{e}(D)$ представляются в следующем виде:

$$\tilde{e}(D) = (\overline{\tilde{e}`(d_1), \ldots, \tilde{e}'(d_{i-1}), \tilde{e}'(d_i), \tilde{e}'(d_{i+1}), \ldots, \tilde{e}'(d_r)}),$$

$$\tilde{o}(D) = \tilde{o}(d_i) = (\overline{\tilde{e}'(d_1), \ldots, \tilde{e}'(d_{i-1}), \tilde{o}'(d_i), \tilde{e}'(d_{i+1}) \ldots, \tilde{e}`(d_r)}),$$

где $i = \overline{1,r}$. Как видно из построения, $\tilde{o}(D)$ и $\tilde{e}(D)$ отличаются в тех же переменных, что и $\tilde{o}'(d_i)$ и $\tilde{e}'(d_i)$. Но по предположению индукции, два последних набора отличаются в значении только одной переменной, что и доказывает утверждение.

Утверждение 5. *Дерево D и поддеревья $d_1, \ldots, d_r$, инцидентные помеченному & ($\vee$) корню, имеют общий набор $\tilde{e}$ ($\tilde{o}$): $\tilde{e}(D) = \tilde{e}(d_i), i = \overline{1,r}$ ($\tilde{o}(D) = \tilde{o}(d_i), i = \overline{1,r}$).*

Доказательство. Следует из способа построения.

Следствие 5.1. *$\tilde{e}(D)$ ($\tilde{o}(D)$) отличается от каждого набора из $\{\tilde{o}(d_{i_1}), \ldots, \tilde{o}(d_{i_s})\}$ ($\{\tilde{e}(d_{i_1}), \ldots, \tilde{e}(d_{i_s})\}$), где $s \leqslant r$, а $d_{i_1}, \ldots, d_{i_s}$ – поддеревья высоты больше 1 и инцидентные корню, в значении ровно одной переменной.*

Доказательство. Следует из утв. 4 и предыдущего утверждения.

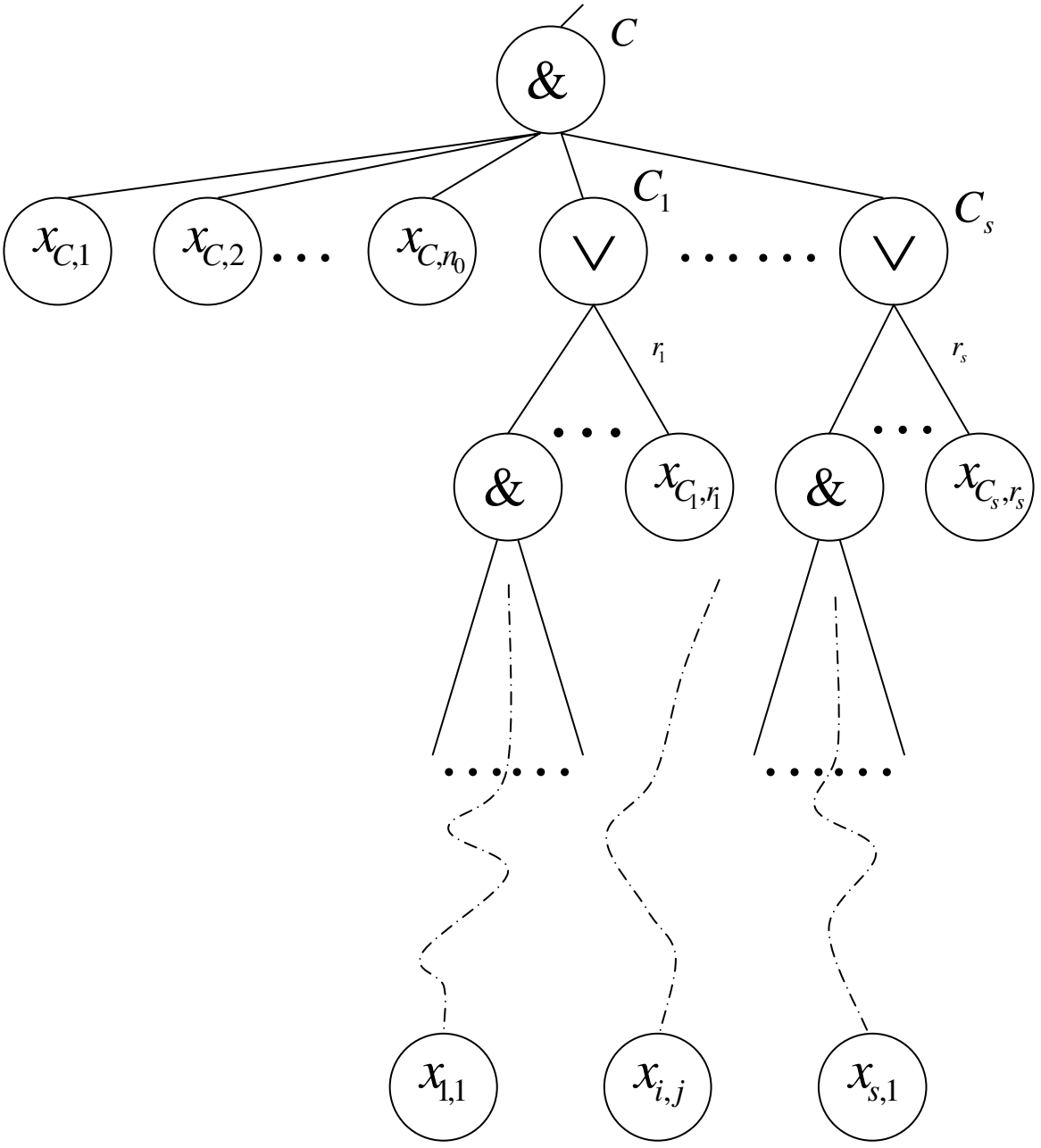
Утверждение 6. *Для любой бесповторной функции f, существенно зависящей от хотя бы одной переменной, множество наборов $S(f)$ является тестом относительно бесповторной альтернативы этой функции.*

Доказательство.

В [6] описан способ получения теста для произвольной бесповторной в B_0 функции от n переменных, длина которого меньше $3.5n$ и это доказательство является, по сути, его модификацией.

Проведем доказательство по индукции. Для деревьев высоты 1 и 2 утверждение проверяется непосредственно. Пусть утверждение справедливо для произвольного канонического дерева высоты не более h. Пусть для заданного дерева построено множество наборов $S(D)$. Так как в это множество входят наборы из $S(d)$ произвольного поддерева d дерева D, которые не отличаются в значениях переменных, от которых реализуемая в d функция не зависит, то по предположению индукции восстановили все поддеревья исходного дерева высоты не более h и не менее 2.

Рассмотрим некоторое каноническое дерево D высоты $h+1$, реализующее бесповторную монотонную функцию f. Пусть C – корневая вершина. Без потери общности будем считать, что вершина помечена символом &. Для случая, когда вершина помечена символом $\vee$ доказательство проводится аналогично.


C
&
C_1
C_s
x_{C,1}
x_{C,2}
x_{C,n_0}
∨
∨
r_1
r_s
&
x_{C_1,r_1}
&
x_{C_s,r_s}
x_{1,1}
x_{i,j}
x_{s,1}

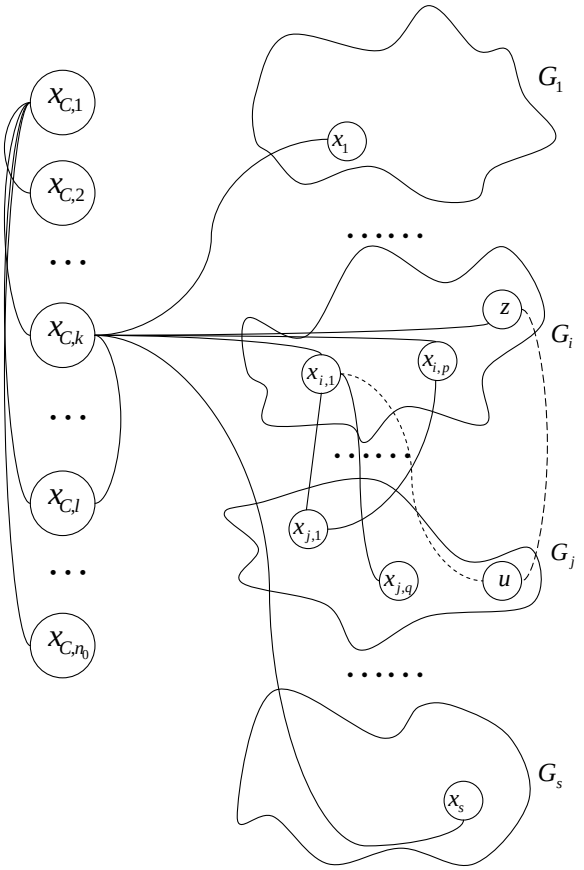


Рис. 6: Граф $G_{\&}$.

Пусть вершина C дугами от корня соединена с s внутренними вершинами $C_1, \ldots, C_s$ и n_0 висячими, как показано на рисунке 5.

Рассмотрим НН-граф $G_\&$, соответствующий поддереву с корнем в вершине C. Для восстановления полной информации о каноническом поддереве достаточно установить монотонность функции и восстановить его граф $G_\&$.

Итак, установим монотонность f, применив исходную индукцию. Так как восстановили все деревья высоты 2, то установили и монотонность всех функций, которыми помечены листовые вершины этих деревьев. Остается доказать монотонность по переменным $\{x_1, \ldots, x_{n_0}\}$, которые соответствуют поддеревьям высоты 1. Докажем, например, монотонность по x_1.

Будем обозначать через $\tilde{\alpha} \leqslant \tilde{\beta}$ ($\tilde{\alpha} \geqslant \tilde{\beta}$), где $\tilde{\alpha}$ и $\tilde{\beta}$ – произвольные наборы булева куба, отношение частичного порядка "меньше либо равно"("больше либо равно") на этом кубе.

Изменим $\tilde{o}(x_1)$ на набор $\tilde{\gamma}_{x_1}$, который получается заменой единиц на нули на местах переменных, в которых $\tilde{o}(x_1)$ отличается от $\tilde{e}(x_1)$, кроме самой переменной x_1. Заметим, что $\tilde{\gamma}_{x_1} \leqslant \tilde{o}(x_1)$, откуда следует $f(\tilde{\gamma}_{x_1}) = 0$. Так как $\tilde{\gamma}_{x_1}$ отличается от $\tilde{e}(x_1)$ в значении переменной x_1 и $f(\tilde{e}(x_1)) = 1$, получаем тест функции x_1, что показывает монотонность исходной функции по x_1. Итак, монотонность функции f полностью доказана.

Теперь перейдем к восстановлению графа $G_\&$. Рассмотрим поддерево d_{x_1}, состоящее из вершины, помеченной символом переменной x_1. Заметим, что в $\tilde{o}(d_{x_1})$ можно уменьшить значения переменных и при этом значение функции сохранится (ноль). Выберем произвольную переменную z, не входящую в $\{x_1, \ldots, x_{n_0}\}$ и рассмотрим поддерево d_z с корнем в вершине, помеченной z. Пусть эта вершина инцидентна вершине, помеченной символом &.

Изменив в наборе $\tilde{o}(d_z)$ значение переменных с 1 на 0, в которых $\tilde{o}(d_z)$ отличается от $\tilde{e}(d_z)$, получим набор $\tilde{\varphi}_z$. Уменьшив в $\tilde{o}(x_1)$ значение переменных с 1 на 0, в которых $\tilde{o}(x_1)$ отличается от $\tilde{e}(d_z)$, получим $\tilde{\varphi}_x$. Заметим, что $\tilde{\varphi}_z \leqslant \tilde{o}(d_z) = 0$ и $\tilde{\varphi}_x \leqslant \tilde{o}(d_{x_1}) = 0$, откуда следует, что $f(\tilde{\varphi}_z) = 0$ и $f(\tilde{\varphi}_x) = 0$. Учитывая, что $f(\tilde{e}(d_z)) = 1$, а наборы $\tilde{\varphi}_z, \tilde{\varphi}_x, \tilde{e}(d_z)$ отличаются только значением переменных x_1 и z, эти наборы образуют тест для остаточной функции $x_1 \& z$.

Если вершина помеченная z инцидентна вершине, помеченной символом $\vee$, поступаем аналогично.

Итак, существуют остаточные функции вида $x_i \& z$, что дает информацию о ребрах, соединяющих вершины переменных $x_1, \ldots, x_{n_0}$ с произвольной другой вершиной в графе $G_\&$. На рисунке 6 схематично изображен граф $G_\&$ и подграфы $G_1, \ldots, G_s$, соответствующие поддеревьям с корнями в вершинах $C_1, \ldots, C_s$ соответственно. Аналогично, наборы $\tilde{o}(x_1), \ldots, \tilde{o}(x_{n_0})$ вместе с единичным набором, на котором функция принимает значение 1, дают информацию об остаточной функции $x_1 \& \ldots \& x_{n_0}$.

Рассмотрим теперь поддеревья $\{d_{1,1}, \ldots, d_{1,r_1}, \ldots, d_{s,1}, \ldots, d_{s,r_s}\}$, соединенные ребрами с соответствующими вершинами $C_1, \ldots, C_s$, и выделим переменные

$$\{x_{1,1}, \ldots, x_{1,r_1}, \ldots, x_{s,1}, \ldots, x_{s,r_s}\}$$

по следующему правилу: если $d_{i,j}$ имеет высоту больше 1, то $\tilde{o}(d_{i,j})$ отличается от $\tilde{e}(d_{i,j})$ в значении единственной переменной $x_{i,j}$ (см. утв. 4), иначе берется переменная, которой помечена единственная в этом поддереве вершина.

Обозначим через d_{C_i} поддерево с корнем в вершине C_i. Согласно следствию 5.1, если поддерево $d_{i,j}$ имеет высоту больше 1, то $\tilde{o}(d_{C_i})$ отличается от $\tilde{e}(d_{i,j})$ в значении ровно одной переменной – $x_{i,j}$. Если же высота $d_{i,j}$ равна 1, то будем вместо $\tilde{e}(d_{i,j})$ рассматривать набор, который получается из $\tilde{e}(d_{C_i})$ заменой $x_{i,j}$ с 0 на 1. Заметим, что на получившимся наборе функция f принимает значение 1, в силу монотонности функции f и того, что получившийся набор отличается от $\tilde{e}(d_{i,j})$ заменой нулей на единицы в некоторых позициях. Будем обозначать наборы $\tilde{e}(d_{i,j})$ или их замены через $\tilde{\alpha}_{i,j}$. Все $\tilde{\alpha}_{i,j}$ отличаются от $\tilde{e}(d_{C_i})$, где $j = \overline{1, r_i}$, в значении ровно одной переменной.

Фиксируем значения переменных, от которых f не зависит, а значения оставшихся переменных, не входящих во множество $\{x_{1,1}, \ldots, x_{1,r_1}, \ldots, x_{s,1}, \ldots, x_{s,r_s}\}$, так же, как в наборе $\tilde{e}(D)$. Заметим, что при этом реализуется остаточная подфункция $\tilde{f}(x_{1,1}, \ldots, x_{1,r_1}, \ldots, x_{s,1}, \ldots, x_{s,r_s})$, существенно зависящая от всех своих переменных. Заметим, что наборы $\tilde{\alpha}_{i,j}$ являются ее нижними единицами. В свою очередь, на наборах $o(d_{C_i})$, соответствующим вершинам $C_1, \ldots, C_s$,

эта функция принимает значение 0, поэтому аналогично предыдущему этапу доказательства устанавливается, что функция $\tilde{f}$ имеет остаточные функции $x_{1,1}$ & $x_{2,1}$ & ... & $x_{s,1}$, $x_{1,1}$ & $x_{2,1}$ & ... & $x_{i-1,1}$ & $x_{i,j}$ & $x_{i+1,1}$ & ... & $x_{s,1}$ для $i = \overline{1,s}; j = \overline{2,r_i}$. Таким образом установили информацию о ребрах, соединяющих вершины, соответствующие парам переменных $(x_{i,1}, x_{j,1})$, $(x_{i,p}, x_{j,1})$, $(x_{i,1}, x_{j,q})$, для $p \neq 1$, $q \neq 1$ (см. рис. 6).

Покажем, что существует ребро $(x_{i,1}, u)$, где путь из корня в переменную u лежит через вершину C_j и ее q-ю дугу, где $q > 1$ (см. рис. 6). Рассмотрим четверку вершин, помеченных $x_{i,1}, x_{j,1}, x_{j,q}, u$. Для начала заметим, что ребер $(x_{j,1}, u)$ и $(x_{j,1}, x_{j,q})$ нет. Если ребро $(x_{j,q}, u)$ есть, получаем по правилу пирамиды (утв. 3) ребро $(x_{i,1}, u)$.

Предположим, что ребер $(x_{j,q}, u)$ и $(x_{i,1}, u)$ нет, тогда рассмотрим дополнительную переменную u_1, отличную от u, такую, что ближайшая общая вершина $x_{j,q}$ и u_1 есть $C_{j,q}$. Тогда по правилу пирамиды есть ребро $(x_{i,1}, u_1)$. Заметим, что есть ребро (u, u_1), поэтому четверка $x_{i,1}, x_{j,1}, u, u_1$ не образует НН-графа, пришли к противоречию с утверждением 2.

Таким же образом можно показать наличия ребер вида $(x_{j,1}, z)$, где z есть произвольная переменная, которой помечена вершина, путь от корня к которой проходит через вершину C_i. Рассматривая четверку $x_{i,1}, x_{j,1}, z, u$, по правилу пирамиды получаем, что между вершинами u и z есть ребро. Итак, определили наличие всех недостающих ребер и полностью восстановили информацию о графе $G_{\&}$. Утверждение доказано.

Лемма 1. *Для произвольной бесповторной функции $f = f(x_1, \ldots, x_n)$ выполняется*

$$T_{B_0}(f) \leqslant n + 1.$$

Доказательство. База индукции. Для функций реализуемых каноническими деревьями высоты 1 утверждение выполняется: $2 = n + 1$.

Индуктивный переход. Пусть верно для всех деревьев высоты не более n. Если корневая вершина помечена символом & ($\vee$), то на последнем этапе построения в результирующего множестве окажется r одинаковых единиц (нулей), поэтому его мощность не превысит $(n_1 + 1) + \ldots + (n_r + 1) - (r - 1) = n_1 + \ldots + n_r + 1 = n + 1$, где $n_1, \ldots, n_r$ – число листовых вершин в соответствующих поддеревьях. Лемма доказана.

Следствие 6.1.

$$T_{B_0}(n) = n + 1.$$

Доказательство. Для функций-констант достаточно одного тестового набора. Для функции $f(\tilde{x}_n)$, существенно зависящей от хотя бы одной переменной, строится $S(f)$, мощность которого не превосходит $n + 1$. Учитывая неравенство $T_{B_0}(n) \geqslant n + 1$ (см. утверждение 1), получаем требуемое. Следствие доказано.

Автор выражает искреннюю благодарность Андрею Анатольевичу Вороненко и Дмитрию Сергеевичу Романову за большое внимание и незаурядное терпение, проявленные во время подготовки настоящей работы.

Список литературы

[1] Чегис И. А., Яблонский С.В. Логические способы контроля работы электрических схем // труды МИ АН СССР. 1958. Т. 51. С. 270–360.

[2] Чегис И. А., Яблонский С.В. О тестах для электрических схем. Усп. мат. наук, 1955, 10, вып. 4 (66), С. 182–184.

[3] Избранные вопросы теории булевых функций. (Под ред. С. В. Винокурова и Н. А. Перязева.) – М.: Физматлит, 2001.

[4] Вороненко А. А. О проверяющих тестах для бесповторных функций // Математические вопросы кибернетики. Вып.11. – М.: Физматлит, 2002. С. 163–176.

[5] Черухин Д. Ю. Алгоритмический критерий сравнения булевых базисов // Математические вопросы кибернетики. Вып.8. – М.: Физматлит, 1999г. С. 77–122.

[6] Вороненко А. А. О длине проверяющего теста для бесповторных функций в базисе $\{0, 1, \&, \vee, \neg\}$ // Дискретная математика. Т.17, Вып.2. – М.: Физматлит, 2005. С. 139–143.

[7] Вороненко А. А. Об оценке длины проверяющего теста для некоторых бесповторных функций. // Прикладная математика и информатика. (2003) 15, С. 85–97.

УДК 519.172.1

О ЧИСЛЕ МАКСИМАЛЬНЫХ НЕЗАВИСИМЫХ МНОЖЕСТВ В ДЕРЕВЬЯХ ФИКСИРОВАННОГО ДИАМЕТРА

© 2009 г. А. Б. Дайняк

dainiak@gmail.com

Кафедра Математической кибернетики

1 Введение

Всякое подмножество попарно не смежных вершин графа называется *независимым*. Под *максимальными независимыми множествами* (м. н. м.) будем понимать максимальные по включению независимые множества. Через $i_M(G)$ будем обозначать число максимальных независимых множеств в G. Множества вершин и рёбер графа G обозначаются через $V(G)$ и $E(G)$ соответственно.

Пусть $d, n \in \mathbb{N}$, и пусть $d < n$. Всякое дерево диаметра d на n вершинах, имеющее максимальное число м. н. м. среди всех деревьев с данным числом вершин и диаметром, будем называть $(n,d)_{\text{м.н.м.}}$*-максимальным*. Расстоянием от вершины $v \in V(G)$ в графе G до подграфа G' будем называть наименьшее из расстояний от v до вершин из $V(G')$. *Диаметром* дерева T будем называть величину $\mathrm{diam}(T)$, равную максимальному числу рёбер в цепи в T (соответствующая цепь называется *диаметральной*).

Первые работы по перечислению независимых множеств в деревьях относятся к 1980х годов. В работе [4] были получены достижимые верхние и нижние оценки числа независимых множеств в деревьях на заданном числе вершин. В [6] была получена достижимая верхняя оценка числа м. н. м. в деревьях с заданным числом вершин, позже в [5] было получено описание экстремальных деревьев. В настоящей работе рассматривается вопрос о числе м. н. м. в деревьях фиксированного диаметра. Аналогичная проблема оценки числа всех (не только максимальных) множеств исследовалась Педерсеном и Вестергардом в [3], где была получена достижимая верхняя оценка. Достижимые нижние оценки числа независимых множеств в общем случае до сих пор не найдены, хотя некоторый прогресс в этом направлении был достигнут в [1, 2].

Приведённая ниже теорема 1 даёт достижимую верхнюю оценку числа м. н. м. в деревьях с заданным числом вершин и диаметром, и описывает соответствующие экстремальные деревья с точностью до изоморфизма. Из теоремы 1 как частные случаи следуют теоремы Вильфа [6] и Сагана [5].

Пусть $U = \{u_1, \ldots, u_{d-1}\}$, $V = \{v_1, \ldots, v_p\}$, $W = \{w_1, \ldots, w_q\}$. Обозначим через $B_{d,p,q}$ дерево диаметра d на множестве вершин $U \cup V \cup W$, такое, что его поддеревья, порождённые множествами $\{u_1\} \cup V$, $\{u_{d-1}\} \cup W$ и U представляют собой соответственно звёзды $K_{1,p}$, $K_{1,q}$ и цепь P_{d-1}. Деревья $B_{d,p,q}$ называются *мётлами (brooms)* (см., например, [3]).

Висячей вершиной (листом) в графе называется вершина степени 1. *Висячим ребром* в дереве будем называть произвольное ребро, инцидентное висячей вершине.

Будем говорить, что граф G' получен *подразбиением* ребра $e = (v_1, v_2)$ графа G, если $V(G') = V(G) \cup \{v'\}$ и $E(G') = (E(G) \setminus \{e\}) \cup \{\{v_1, v'\}, \{v_2, v'\}\}$.

Пусть T — произвольное дерево, а T' — его поддерево. Пусть $v \in V(T) \setminus V(T')$, $u \in V(T')$. Будем говорить, что дерево T' *примыкает* вершиной u к вершине v в T, если $\{u, v\} \in E(T)$ и дерево T' является компонентой связности леса $T \setminus \{v\}$. Например, можно сказать, что во всякой цепи чётного диаметра $2d$ к центральной вершине примыкают своими концевыми вершинами ровно две цепи диаметра $(d-1)$.

Введём несколько обозначений для деревьев специального вида. Через $\widetilde{B}_{4,n}$ обозначим дерево, полученное из звезды $K_{1,\frac{n-1}{2}}$ подразбиением всех рёбер. Через $\widetilde{B}'_{4,n}$ обозначим дерево,

полученное из звезды $K_{1,\frac{n}{2}}$ подразбиением $\frac{n-2}{2}$ рёбер. Через $\widetilde{B}'_{5,n}$ обозначим дерево, полученное из $B_{3,\frac{n-5}{2},2}$ подразбиением всех висячих рёбер, кроме одного ребра, инцидентного вершине степени 3. Через $\widetilde{B}_{5,n,p}$ обозначим дерево, полученное из $B_{3,p,\frac{n-5-2p}{2}}$ подразбиением всех висячих рёбер. Через $\widehat{B}'_{5,n,p}$ обозначим дерево, получающееся присоединением висячей вершины к той из центральных вершин дерева $\widetilde{B}_{5,n-1,p}$, которая имеет степень $(p+1)$. Через $\widehat{B}''_{5,n,p}$ обозначим дерево, получающееся присоединением висячей вершины к каждой из центральных вершин дерева $\widetilde{B}_{5,n-2,p}$. Через $\widehat{B}_{6,n,p}$ обозначим дерево, полученное из $B_{4,p,\frac{n-3-2p}{2}}$ подразбиением всех висячих рёбер. Через $\widetilde{B}'_{6,n,p}$ обозначим дерево, получающееся присоединением висячей вершины к центральной вершине дерева $\widehat{B}_{6,n-1,p}$. Через $\widehat{B}_{6,n,p,q}$ обозначим дерево, полученное присоединением q висячих вершин к центру $B_{4,p,\frac{n-3-2p-2q}{2}}$ и последующим подразбиением всех висячих рёбер. Через $\widehat{B}'_{6,n,p,q}$ обозначается дерево, полученное присоединением висячей вершины к центральной вершине дерева $\widehat{B}_{6,n-1,p,q}$. Через $\widetilde{B}'_{7,n}$ обозначим дерево, полученное из $B_{5,\frac{n-7}{2},2}$ подразбиением всех висячих рёбер, кроме одного ребра, инцидентного вершине степени 3. Через $\widetilde{B}_{7,n}$ обозначим дерево, полученное из $B_{5,p,\frac{n-4-2p}{2}}$ подразбиением всех висячих рёбер. Через $\widehat{B}_{7,n,p,q,r}$ обозначим дерево, полученное присоединением к центральным вершинам дерева $B_{5,p,\frac{n-5-2p-2q-2r}{2}}$, смежным с вершинами степени $(p+1)$ и $\frac{n-3-2p-2q-2r}{2}$, соответственно $(q+1)$ и r висячих вершин, и последующим подразбиением всех висячих рёбер, кроме одного, инцидентного центральной вершине, смежной с вершинами степени $(p+1)$ и $(r+2)$. Обозначим через $\widehat{B}_{8,n}$ дерево, полученное из $\widetilde{B}_{4,n-4}$ двукратным подразбиением двух висячих рёбер. Через $\widehat{B}'_{8,n}$ обозначим дерево, полученное присоединением висячей вершины к центру дерева $\widehat{B}_{8,n-1}$. Через $\widehat{B}_{9,n,p}$ обозначим дерево, полученное из дерева $\widetilde{B}_{5,n-4,p+1}$ двукратным подразбиением двух висячих рёбер, находящихся на противоположных концах дерева. Через $\widehat{B}'_{9,n,p}$ обозначим дерево, полученное присоединением висячей вершины к центральной вершине степени $(p+2)$ дерева $\widehat{B}_{9,n-1,p}$. Через $\widehat{B}''_{9,n,p}$ обозначим дерево, полученное присоединением висячих вершин к каждой из центральных вершин степени дерева $\widehat{B}_{9,n-2,p}$. Через $\widetilde{B}'_6$ обозначим дерево, полученное из $K_{1,3}$ двукратным подразбиением двух рёбер. Через $\widetilde{B}'_8$ обозначим дерево, полученное присоединением висячей вершины к четвёртой с конца вершине цепи P_9. Через $\widetilde{B}_d$ обозначим дерево, полученное присоединением висячей вершины к третьей с конца вершине цепи P_{d+1}. Через $\check{B}_{d,n}$ обозначим дерево, полученное из $B_{d-2,\frac{n-d+1}{2},1}$ подразбиением всех висячих рёбер. Через $\check{B}'_{d,n}$ обозначим дерево, полученное присоединением висячей вершины к той вершине дерева $\check{B}_{d,n-1}$, которая не смежна с висячими, но смежна с вершиной степени $\frac{n-d}{2}$. Наконец, через $\widehat{B}^*_{d,k}$ обозначим дерево, полученное присоединением висячей вершины к k-й вершине диаметральной цепи дерева дерева $\check{B}_{d,d+3}$, считая с того конца диаметральной цепи, который наиболее удалён от вершины степени 3.

2 Вспомогательные утверждения

Обозначим через ψ_n число м. н. м. в цепи на n вершинах. Последовательность ψ_n, очевидно, удовлетворяет соотношению $\psi_n = \psi_{n-2} + \psi_{n-3}$ и начальным условиям $\psi_0 = \psi_1 = 1$, $\psi_2 = 2$. Значения чисел ψ_n для небольших n приведены в табл. 1.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ψ_n	1	1	2	2	3	4	5	7	9	12	16	21	28	37	49	65

Таблица 1: Значения ψ_n

Утверждение 1. *Пусть T — произвольное дерево. Пусть в T есть вершина, смежная с двумя или более листьями, и пусть u — один из этих листьев. Тогда для дерева T', полученного из T удалением вершины u, выполнено равенство $i_M(T') = i_M(T)$.*

Доказательство. Достаточно заметить, что если $u_1, \ldots, u_r$ — листья, имеющие общего соседа в T, то во всякое м. н. м. в T либо входят одновременно все вершины $u_1, \ldots, u_r$, либо не входит ни одна из них. □

Лемма 1. *При любых n и d таких, что $4 \leqslant d < n$, в $(n,d)_{\text{м.н.м.}}$-максимальных деревьях каждая вершина смежна не более чем с одним листом.*

Доказательство. Предположим, что $d \geqslant 4$ и найдётся $(n,d)_{\text{м.н.м.}}$-максимальное дерево T, в котором есть вершина, смежная с двумя и более листьями. Удалив один из этих листьев, получим дерево T', для которого, по утверждению 1, выполнено равенство $i_M(T') = i_M(T)$. Кроме того, очевидно, $\operatorname{diam}(T') = \operatorname{diam}(T)$ и $|V(T')| = |V(T)| - 1$. Во всяком дереве, диаметр которого не меньше четырёх, найдётся вершина, либо не смежная ни с одним из листьев, либо являющаяся листом, не лежащим на диаметральной цепи. Пусть v — такая вершина в дереве T'. Добавив в T' новую листовую вершину u, соединив её с v; тем самым получим дерево T'', для которого $|V(T'')| = |V(T)|$, $\operatorname{diam}(T'') = \operatorname{diam}(T)$ и $i_M(T'') > i_M(T)$. Но это противоречит выбору T как $(n,d)_{\text{м.н.м.}}$-максимального дерева. Полученное противоречие завершает доказательство. □

Для натуральных n, d таких, что $4 \leqslant d \leqslant n-1$, определим величину $M(n,d)$:

$$M(n,d) = \begin{cases} \psi_{d-1} + (2^{(n-d+1)/2} - 1)\psi_{d-2}, & \text{при } d \geqslant 4,\ n-d = 2k+1,\ k \geqslant 0, \\ \psi_{d-2} + \psi_d, & \text{при } d \geqslant 4,\ n-d = 2, \\ 2^{(n-d)/2}\psi_{d-1}, & \text{при } d \geqslant 5,\ d \neq 7,\ n-d = 2k \geqslant 4, \\ 2^{(n-d)/2}\psi_{d-1} + 1, & \text{при } d \in \{4,7\},\ n-d = 2k \geqslant 4. \end{cases}$$

Утверждение 2.

1. *При $d \geqslant 4$ и любом n, $n \geqslant d+3$, таком, что $2 \nmid (n-d)$, выполнено неравенство $M(n,d) > M(n,d+1)$.*
2. *При $d \geqslant 4$ и любом n, $n \geqslant d+2$, таком, что $2 \mid (n-d)$, выполнено неравенство $M(n,d) \leqslant M(n,d+1)$, причём $M(n,d) = M(n,d+1)$ только если $d = 4$.*
3. *При $d \geqslant 4$ и любом n, $n \geqslant d+3$, имеет место неравенство $M(n,d) \geqslant M(n,d+2)$, причём равенство $M(n,d) = M(n,d+2)$ выполнено только если одновременно $d = 5$ и n чётно.*

Доказательство.

1. Пусть $4 \leqslant d \leqslant n-3$ и $2 \nmid (n-d)$. Если $n = d+3$, то
$$M(n,d) - M(n,d+1) = 2\psi_{d-2} - \psi_{d-1} > 0.$$
Если $n \geqslant d+5$ и $d \neq 6$, то
$$M(n,d) - M(n,d+1) = \psi_{d-1} - \psi_{d-2} + 2^{(n-d-1)/2}(2\psi_{d-2} - \psi_d) \geqslant \psi_{d-1} + 7\psi_{d-2} - 4\psi_d > 0.$$
Если $n \geqslant d+5$ и $d = 6$, то $M(n,d) - M(n,d+1) = 2^{(n-7)/2} > 0$.
2. При $d = 4$ и чётном n равенство $M(n,d) = M(n,d+1)$ легко проверяется. Пусть $5 \leqslant d \leqslant n-2$ и $2 \mid (n-d)$. Если $n = d+2$, то $M(n,d+1) - M(n,d) = \psi_{d-1} - \psi_{d-2} > 0$. Если $d \neq 7$ и $n \geqslant d+4$, то $M(n,d+1) - M(n,d) = \psi_d - \psi_{d-1} > 0$. Если $d = 7$ и $n \geqslant d+4$, то $M(n,d+1) - M(n,d) = 1 > 0$.

3. При $4 \leqslant d \leqslant n-3$ и $2 \nmid (n-d)$ имеем $M(n,d) - M(n,d+2) = (2^{(n-d-1)/2} - 1)(2\psi_{d-2} - \psi_d)$, откуда следует, что $M(n,5) = M(n,7)$, и $M(n,d) > M(n,d+2)$ при $d \neq 5$.

   Если $d = 4$ и n чётно, то $M(n,d) - M(n,d+2) = 1 > 0$. Пусть $5 \leqslant d \leqslant n-3$. Если $n = d+4$, то $M(n,d) - M(n,d+2) \geqslant 3\psi_{d-1} - 2\psi_d > 0$. Если $n \geqslant d+6$ и $2 \mid (n-d)$, то

$$M(n,d) - M(n,d+2) \geqslant 2^{(n-d-2)/2}(2\psi_{d-1} - \psi_{d+1}) - 1 > 0.$$

□

Из утверждения 2 непосредственно вытекает следующий факт.

Лемма 2. *Если* $4 \leqslant d' < d'' \leqslant n-1$, *то*

1. $$\operatorname*{Argmax}_{d' \leqslant d \leqslant d''} M(n,d) = \begin{cases} \{d'+1\}, & \text{при } d' \geqslant 5,\ 2 \mid (n-d'), \\ \{4,5,7\} \cap [d', d''], & \text{при } d' \leqslant 5,\ 2 \mid n, \\ \{d'\}, & \text{при } d' \geqslant 4,\ d' \neq 5,\ 2 \nmid (n-d'). \end{cases}$$

2. $$\max_{d' \leqslant d \leqslant d''} M(n,d) = \begin{cases} M(n,d'+1), & \text{при } 2 \mid (n-d'), \\ M(n,d'), & \text{при } 2 \nmid (n-d'). \end{cases}$$

3. $\max\limits_{d \geqslant d'} M(n,d) \leqslant M(n,4)$.

Лемма 3. *При* $n-2 = d \geqslant 5$ *всякое* $(n,d)_{\text{м.н.м.}}$*-максимальное дерево* T *изоморфно одному из деревьев* $\widetilde{B}^*_6$, $\widetilde{B}^*_8$, $\widetilde{B}''_n$. *При этом* $i_M(T) = M(n,d)$.

Доказательство. Докажем лемму индукцией по d. При $5 \leqslant d \leqslant 8$ справедливость утверждения леммы проверяется перебором. Пусть $d \geqslant 9$ и утверждение леммы выполнено для деревьев диаметра не выше $(d-1)$. Пусть T — $(d+2,d)_{\text{м.н.м.}}$-максимальное дерево. Пусть v — единственная не лежащая на диаметральной цепи T вершина. Обозначим через u ту концевую вершину диаметральной цепи, которая находится от v на наибольшем расстоянии. Пусть u' — соседняя с u вершина, а u'' — вершина, находящаяся от u на расстоянии 2. Тогда, пользуясь предположением индукции, можно оценить $i_M(T)$:

$$i_M(T) = i_M(T \setminus \{u,u'\}) + i_M(T \setminus \{u,u',u''\}) \leqslant M(n-2,d-2) + M(n-3,d-3) = M(n,d),$$

причём равенство $i_M(T) = M(n,d)$ возможно только если деревья $T \setminus \{u,u'\}$ и $T \setminus \{u,u',u''\}$ изоморфны каким-либо из деревьев $\widetilde{B}^*_6$, $\widetilde{B}^*_8$, $\widetilde{B}''_{n-2}$, $\widetilde{B}''_{n-3}$. Но тогда и само дерево T изоморфно дереву $\widetilde{B}''_n$. □

Лемма 4. *Всякое* $(n,5)_{\text{м.н.м.}}$*-максимальное дерево изоморфно одному из деревьев* $\widetilde{B}'_{5,n}$, $\widetilde{B}_{5,n,p}$ *(при некотором* p*).*

Доказательство. Пусть T — произвольное $(n,5)_{\text{м.н.м.}}$-максимальное дерево. Если дерево T изоморфно одному из деревьев $\widetilde{B}'_{5,n}$ или $\widetilde{B}_{5,n,p}$, то, как легко проверить, $i_M(T) = M(n,5)$. Допустим, T не изоморфно ни одному из деревьев $\widetilde{B}'_{5,n}$, $\widetilde{B}_{5,n,p}$. Тогда из леммы 1 следует, что возможны только следующие случаи:

1. T имеет вид $\widehat{B}'_{5,n,p}$, где $n \geqslant 9$ и $p \geqslant 2$. Тогда

$$i_M(T) = 2^{(n-5)/2}(2 + 2^{1-p}) < 3 \cdot 2^{(n-5)/2} = M(n,5).$$

2. T имеет вид $\widehat{B}''_{5,n,p}$, где $n \geqslant 8$. Тогда

$$i_M(T) = 2^{(n-4)/2} + 2^p + 2^{(n-2p-4)/2} \leqslant 2 + 3 \cdot 2^{(n-6)/2} < 1 + 4 \cdot 2^{(n-6)/2} = M(n,5).$$

Таким образом, в обоих рассмотренных случаях мы получаем противоречие с $(n,5)_{\text{м.н.м.}}$-максимальностью T, что завершает доказательство леммы. □

3 Основная теорема

Очевидно, что всякое дерево диаметра 1 или 2 содержит два м. н. м., а всякое дерево диаметра 3 содержит три м. н. м. При $d \geqslant 4$ полное описание $(n,d)_{\text{м.н.м.}}$–максимальных деревьев даётся следующей теоремой.

Теорема 1. *При $4 \leqslant d \leqslant n-2$ для всякого $(n,d)_{\text{м.н.м.}}$–максимального дерева T выполнено равенство $i_M(T) = M(n,d)$, и при этом само дерево T изоморфно одному из деревьев, приведённых в таблице:*

d	$(n-d)$	*экстремальные деревья*
4	$2k+1$ $(k \geqslant 1)$	$\widetilde{B}_{4,n}$
4	$2k$ $(k \geqslant 1)$	$\widetilde{B}'_{4,n}$
5	$2k+1$ $(k \geqslant 1)$	$\widetilde{B}_{5,n,p}$ $(1 \leqslant p \leqslant \frac{n-4}{2})$
5	$2k$ $(k \geqslant 1)$	$\widetilde{B}'_{5,n}$
6	$2k+1$ $(k \geqslant 1)$	$\widetilde{B}_{6,n}$
6	2	$\widetilde{B}^*_6$, $\widetilde{B}''_n$
6	$2k$ $(k \geqslant 2)$	$\widetilde{B}'_{6,n,p}$ $(1 \leqslant p \leqslant \frac{n-6}{2})$
7	$2k+1$ $(k \geqslant 1)$	$\widetilde{B}_{7,n}$ $(1 \leqslant p \leqslant \frac{n-6}{2})$
7	$2k$ $(k \geqslant 1)$	$\widetilde{B}'_{7,n}$
8	$2k+1$ $(k \geqslant 1)$	$\widetilde{B}_{8,n}$
8	2	$\widetilde{B}^*_8$, $\widetilde{B}''_n$
8	$2k$ $(k \geqslant 2)$	$\widetilde{B}'_{8,n}$
$\geqslant 9$	$2k+1$ $(k \geqslant 1)$	$\widetilde{B}_{d,n}$
$\geqslant 9$	2	$\widetilde{B}''_n$
$\geqslant 9$	$2k$ $(k \geqslant 2)$	$\widetilde{B}'_{d,n}$

Доказательство. Справедливость теоремы при $d = 4$ непосредственно следует из леммы 1, а при $d = 5$ обеспечивается леммой 4. Кроме того, при $n = d+2$ утверждение теоремы следует из леммы 3.

Пусть $d \geqslant 6$, $n \geqslant d+3$, и пусть утверждение теоремы выполнено для всех пар (n'', d) таких, что $n'' < n$, а также для всех пар (n', d') таких, что $d' \leqslant d-1$. Докажем, что тогда утверждение теоремы справедливо и для пары (n,d). Пусть T — произвольное $(n,d)_{\text{м.н.м.}}$–максимальное дерево, и P — произвольная диаметральная цепь в T. Покажем для любой вершины v, не лежащей на цепи P, минимальное из расстояний от v до вершин P не превосходит 2. Предположим противное и покажем, что в этом предположении выполнено строгое неравенство $M(n,d) - i_M(T) > 0$, что противоречит выбору T. Из леммы 1 следует, что возможны лишь следующие случаи:

A. В T найдётся вершина, к которой примыкают t двухвершинных цепей и дерево T', где $\operatorname{diam}(T') = d$ и $1 \leqslant t \leqslant \frac{n-d-2}{2}$. В этом случае $M(n,d) - i_M(T) \geqslant D_1(n,d)$, где

$$D_1(n,d) = M(n,d) - M(n-2,d) - M(n-2t-1,d) \cdot 2^{t-1}.$$

Рассмотрим несколько подслучаев:

A1. $n = d+4$. Тогда $t = 1$ и $D_1(n,d) \geqslant 4\psi_{d-1} - \psi_{d-2} - \psi_d - \psi_{d+1} > 0$.

A2. $n = d+2k$, где $k \geqslant 3$. Тогда

$$\begin{aligned} D_1(n,d) &= 2^{(n-d)/2}\psi_{d-1} - 2^{(n-d-2)/2}(\psi_{d-1} + \psi_{d-2}) - 2^{t-1}(\psi_{d-1} - \psi_{d-2}) \geqslant \\ &\geqslant 2^{(n-d)/2}\psi_{d-1} - 2^{(n-d-2)/2}(\psi_{d-1} + \psi_{d-2}) - 2^{(n-d-4)/2}(\psi_{d-1} - \psi_{d-2}) = \\ &= 2^{(n-d-4)/2}(\psi_{d-1} - \psi_{d-2}) > 0. \end{aligned}$$

A3. $n = d+2k+1$, где $k \geqslant 2$, и $t = \frac{n-d-3}{2}$. Тогда $D_1(n,d) = 2^{(n-d-5)/2}(3\psi_{d-2} - \psi_d) > 0$.

A4. $n = d+2k+1$, где $k \geqslant 3$, и $t \leqslant \frac{n-d-5}{2}$. Тогда $D_1(n,d) \geqslant 2^{(n-d-7)/2}(8\psi_{d-2} - 4\psi_{d-1} - 1) > 0$.

B. В T найдётся вершина, к которой примыкают t двухвершинных цепей, одна висячая вершина, и дерево T', где $\operatorname{diam}(T') = d$ и $1 \leqslant t \leqslant \frac{n-d-3}{2}$. В этом случае $M(n,d) - i_M(T) \geqslant D_2(n,d)$, где $D_2(n,d) = M(n,d) - M(n-2,d) - M(n-2t-2,d) \cdot 2^{t-1}$. Рассмотрим несколько подслучаев:
B1. $n = d + 2k + 1$, где $k \geqslant 2$. Тогда

$$\begin{aligned} D_2(n,d) &= 2^{(n-d-3)/2}\psi_{d-2} - 2^{t-1}(\psi_{d-1} - \psi_{d-2}) \geqslant \\ &\geqslant 2^{(n-d-3)/2}\psi_{d-2} - 2^{(n-d-5)/2}(\psi_{d-1} - \psi_{d-2}) = \\ &= 2^{(n-d-5)/2}(3\psi_{d-2} - \psi_{d-1}) > 0. \end{aligned}$$

B2. $n = d + 2k$, где $k \geqslant 3$ и $t = \frac{n-d-4}{2}$. Тогда $D_2(n,d) = 2^{(n-d-6)/2}(4\psi_{d-1} - \psi_d - \psi_{d-2}) > 0$.
B3. $n = d + 2k$, где $k \geqslant 3$ и $t \leqslant \frac{n-d-6}{2}$. Тогда $D_2(n,d) > 2^{(n-d-4)/2}(\psi_{d-1} - 1) > 0$.

Итак, в любом случае наличие вершин, отстоящих от цепи P на расстояние, большее двух, вступает в противоречие с $(n,d)_{\text{м.н.м.}}$–максимальностью T. Отсюда, и из леммы 1 следует, что всякая вершина в T удалена от диаметральной цепи на расстояние не больше 2, и любая вершина T смежна не более чем с одним листом.

Рассмотрим отдельно случаи, когда дерево T имеет диаметр 6 и 7.
C. $\operatorname{diam}(T) = 6$. Допустим, дерево T не изоморфно ни одному из деревьев $\check{B}_{6,n}$, $\widetilde{B}'_{6,n,p}$. Тогда возможны следующие случаи:
C1. В T найдётся вершина, к которой примыкают t двухвершинных цепей, одна висячая вершина, и дерево T', где $1 \leqslant t \leqslant \frac{n-6}{2}$ и $\operatorname{diam}(T') \geqslant 3$. Если в T' всего четыре вершины, то

$$i_M(T) = 2 + 3 \cdot 2^{(n-6)/2} < M(n,6).$$

Пусть в T' по крайней мере 5 вершин. Тогда из леммы 2 и предположения индукции следует, что

$$M(n,6) - i_M(T) \geqslant M(n,6) - \max_{d\in\{5,6\}} M(n-2,d) - 2^{t-1}M(n-2t-2,4).$$

Если n чётно, то $\max_{d\in\{5,6\}} M(n-2,d) = M(n-2,5)$ и

$$M(n,6) - i_M(T) \geqslant M(n,6) - M(n-2,5) - 2^{t-1}M(n-2t-2,4) \geqslant 2^{(n-8)/2} - 1 > 0.$$

Если n нечётно, то $\max_{d\in\{5,6\}} M(n-2,d) = M(n-2,6)$ и

$$M(n,6) - i_M(T) \geqslant M(n,6) - M(n-2,6) - 2^{t-1}M(n-2t-2,4) = 2^{(n-7)/2} > 0.$$

C2. T либо изоморфно одному из деревьев $\widehat{B}_{6,n,p,q}$, $\widehat{B}'_{6,n,p,q}$, либо изоморфно дереву $\widehat{B}_{6,n,p}$ и при этом $2 \leqslant p \leqslant \frac{n-7}{2}$. Из приведённой ниже таблицы видно, что во всех этих случаях $i_M(T) < M(n,6)$.

T	$i_M(T)$	нижняя оценка для $(M(n,6) - i_M(T))$
$\widehat{B}_{6,n,p,q}$	$2^{(n-3)/2} + 2^p + 2^{(n-3-2p-2q)/2} - 1$	$2^{(n-7)/2}$
$\widehat{B}'_{6,n,p,q}$	$2^{(n-4)/2} + 2^{(n-4-2q)/2}$	$2^{(n-6)/2}$
$\widehat{B}_{6,n,p}$	$2^{(n-3)/2} + 2^p + 2^{(n-3-2p)/2} - 1$	$2^{(n-7)/2} - 2$

D. $\operatorname{diam}(T) = 7$. Возможны следующие подслучаи:
D1. В T найдётся вершина, к которой примыкают t двухвершинных цепей, одна висячая вершина, и дерево T', где $1 \leqslant t \leqslant \frac{n-7}{2}$ и $\operatorname{diam}(T') \geqslant 4$. Тогда из леммы 2 и предположения индукции следует, что

$$M(n,7) - i_M(T) \geqslant M(n,7) - \max_{d\in\{6,7\}} M(n-2,d) - 2^{t-1}M(n-2t-2,4).$$

Если n чётно, то $\max_{d\in\{6,7\}} M(n-2,d) = M(n-2,7)$ и

$$M(n,7) - i_M(T) \geqslant M(n,7) - M(n-2,7) - 2^{t-1}M(n-2t-2,4) \geqslant 3 \cdot 2^{(n-10)/2} > 0.$$

Если n нечётно, то $\max_{d\in\{6,7\}} M(n-2,d) = M(n-2,6)$ и

$$M(n,7) - i_M(T) \geqslant M(n,7) - M(n-2,6) - 2^{t-1}M(n-2t-2,4) = 0,$$

причём из леммы 2 и предположения индукции следует, что равенство $M(n,7) - i_M(T) = M(n,7) - M(n-2,6) - 2^{t-1}M(n-2t-2,4)$ возможно только если $t = 1$ и дерево T' имеет вид $\widetilde{B}_{4,n'}$, а это возможно только если T изоморфно дереву $\widetilde{B}'_{7,n}$.

D2. В T найдётся вершина, к которой примыкают t двухвершинных цепей и дерево T', где $1 \leqslant t \leqslant \frac{n-6}{2}$. Если n чётно, то из предположения индукции и леммы 2 получаем

$$i_M(T) \leqslant M(n-2,5) + 2^{t-1}M(n-2t-1,4) = M(n,7),$$

причём равенство $i_M(T) = M(n,7)$ возможно только если поддерево T' дерева T изоморфно дереву $\widetilde{B}_{4,n'}$, и значит само дерево T изоморфно дереву $\widetilde{B}_{7,n}$.

Допустим теперь, что n нечётно, и T не изоморфно дереву $\widetilde{B}'_{7,n}$, а также не может быть представлено в виде, описанном в п. D1. Тогда T изоморфно дереву $\widehat{B}_{7,n,p,q,r}$ при некоторых $q, r \geqslant 0$, и выполнены соотношения

$$i_M(T) = 2^{(n-5)/2} + 2^{p+q} + 2^{(n-2q-5)/2} \leqslant 5 \cdot 2^{(n-7)/2} < M(n,7).$$

Все случаи, когда $d = \operatorname{diam}(T) \leqslant 7$, разобраны выше, и на протяжении оставшейся части доказательства теоремы мы будем считать, что $d \geqslant 8$. Зафиксируем в дереве T какую-нибудь диаметральную цепь P. Обозначим через w, w', u, u', u'' идущие по порядку вершины P, где w — концевая вершина P. При этом будем считать, что если $\tilde{w}$, $\tilde{w}'$, $\tilde{u}$, $\tilde{u}'$, $\tilde{u}''$ — вершины P, идущие по порядку начиная с концевой вершины $\tilde{w}$, противоположной w, то набор степеней $(\deg w, \deg w', \deg u, \deg u', \deg u'')$ лексикографически не меньше набора степеней $(\deg \tilde{w}, \deg \tilde{w}', \deg \tilde{u}, \deg \tilde{u}', \deg \tilde{u}'')$. Рассмотрим несколько случаев:

1. К вершине u в T примыкают t цепей на двух вершинах ($t \geqslant 2$), и некоторое дерево T', где $\operatorname{diam}(T') \geqslant d-3$. Имеем

$$i_M(T) \leqslant M(n-2,d) + 2^{t-1} \cdot \max_{d' \geqslant d-3} M(n-2t-1,d'). \tag{1}$$

1.1. Если $2 \nmid (n-d)$, то из леммы 2 следует, что

$$\max_{d' \geqslant d-3} M(n-2t-1,d') \leqslant M(n-2t-1,d-3).$$

Тогда (1) влечёт

$$\begin{aligned} M(n,d) \quad & -i_M(T) \geqslant M(n,d) - M(n-2,d) - 2^{t-1} \cdot M(n-2t-1,d-3) = \\ & = 2^{(n-d-1)/2}\psi_{d-2} - 2^{(n-d+1)}\psi_{d-5} - 2^{t-1}(\psi_{d-4} - \psi_{d-5}) \geqslant \\ & \geqslant 2^{(n-d-1)/2}\psi_{d-2} - 2^{(n-d+1)/2}\psi_{d-5} - 2^{(n-d-1)/2}(\psi_{d-4} - \psi_{d-5}) = 0, \end{aligned}$$

причём равенство $i_M(T) = M(n,d)$ имеет место только в случае $t = \frac{n-d+1}{2}$, то есть только если дерево T имеет вид $\check{B}_{d,n}$.

1.2. Если $2 \mid (n-d)$, то из леммы 2 следует, что

$$\max_{d' \geqslant d-3} M(n-2t-1,d') \leqslant M(n-2t-1,d-2).$$

Тогда возможны две ситуации.

1.2.1. $n \geqslant d+6$. Тогда

$$\begin{aligned} M(n,d) - i_M(T) \quad & \geqslant M(n,d) - M(n-2,d) - 2^{t-1} \cdot M(n-2t-1,d-2) = \\ & = (2^{(n-d-2)/2} - 2^{t-1})(\psi_{d-3} - \psi_{d-4}) \geqslant 0, \end{aligned}$$

причём равенство $i_M(T) = M(n,d)$ возможно только если $t = \frac{n-d}{2}$ и $\operatorname{diam}(T') = d-2$. Но тогда дерево T имеет вид $\check{B}'_{d,n}$.

1.2.2. $n = d+4$. Тогда дерево T имеет вид $\widehat{B}^*_{d,k}$ при $k \geqslant 3$. Если T имеет вид $\widehat{B}^*_{d,3}$, то $M(n,d) - i_M(T) = 4\psi_{d-1} - 2(\psi_d + \psi_{d-5}) - \psi_{d-3} > 0$. Если T имеет вид $\widehat{B}^*_{d,4}$, то

$$M(n,d) - i_M(T) = 2\psi_{d-1} - 4\psi_{d-4} > 0.$$

Если же T имеет вид $\widehat{B}^*_{d,k}$, $k \geqslant 5$, то при $8 \leqslant d \leqslant 10$ утверждение теоремы проверяется перебором, а при $d \geqslant 11$ из предположения индукции следует, что

$$M(n,d) - i_M(T) \geqslant M(n,d) - M(n-2,d-2) - M(n-3,d-3) = 0,$$

причём $i_M(T) = M(n,d)$ только если T имеет вид $\widetilde{B}'_{4,n}$.

2. К вершине u в T примыкают t цепей на двух вершинах ($t \geqslant 2$), один лист, и некоторое дерево T', где $\operatorname{diam}(T') \geqslant d-3$. Имеем

$$i_M(T) \leqslant M(n-2,d) + 2^{t-1} \cdot \max_{d' \geqslant d-3} M(n-2t-2,d'). \tag{2}$$

2.1. Если $2 \nmid (n-d)$, то из леммы 2 следует, что

$$\max_{d' \geqslant d-3} M(n-2t-1,d') \leqslant M(n-2t-2,d-2).$$

Тогда (2) влечёт

$$\begin{aligned} M(n,d) - i_M(T) &\geqslant M(n,d) - M(n-2,d) - 2^{t-1} \cdot M(n-2t-2,d-2) = \\ &= 2^{(n-d-1)/2}(\psi_{d-2} - \psi_{d-4}) - 2^{t-1}(\psi_{d-3} - \psi_{d-4}) \geqslant \\ &\geqslant 2^{(n-d-1)/2}(\psi_{d-2} - \psi_{d-4}) - 2^{(n-d-3)/2}(\psi_{d-3} - \psi_{d-4}) = \\ &= 2^{(n-d-3)/2}(2\psi_{d-2} - \psi_{d-1}) > 0. \end{aligned}$$

2.2. Если $2 \mid (n-d)$, то из леммы 2 следует, что

$$\max_{d' \geqslant d-3} M(n-2t-2,d') \leqslant M(n-2t-2,d-3).$$

Тогда (2) влечёт

$$\begin{aligned} M(n,d) \quad -i_M(T) &\geqslant M(n,d) - M(n-2,d) - 2^{t-1} \cdot M(n-2t-2,d-3) = \\ &= 2^{(n-d-2)/2}\psi_{d-1} - 2^{t-1}(\psi_{d-4} - \psi_{d-5}) - 2^{(n-d)/2}\psi_{d-5} \geqslant \\ &\geqslant 2^{(n-d-2)/2}\psi_{d-1} - 2^{(n-d-2)/2}(\psi_{d-4} - \psi_{d-5}) - 2^{(n-d)/2}\psi_{d-5} = 0, \end{aligned}$$

причём для равенства $i_M(T) = M(n,d)$ необходимо, чтобы $t = \frac{n-d}{2}$. Но при $t = \frac{n-d}{2}$ имеем

$$M(n,d) - i_M(T) = 2^{(n-d)/2}(\psi_{d-1} - \psi_{d-2}) - \psi_{d-3} \geqslant 4(\psi_{d-1} - \psi_{d-2}) - \psi_{d-3} > 0.$$

3. К вершине u в T примыкает одна цепь на двух вершинах, один лист, и дерево T', где $\operatorname{diam}(T') \geqslant d-3$. В этом случае, также как и в предыдущих, применяем предположение индукции и лемму 2:

3.1. Если $2 \mid (n-d)$, то

$$\begin{aligned} M(n,d) - i_M(T) &\geqslant M(n,d) - M(n-2,d-1) - M(n-4,d-3) = \\ &= (2^{(n-d)/2} - 1)(\psi_{d-4} - \psi_{d-5}) - \psi_{d-2} + \psi_{d-3} \geqslant \\ &\geqslant 3(\psi_{d-4} - \psi_{d-5}) - \psi_{d-2} + \psi_{d-3} > 0. \end{aligned}$$

3.2. Если $2 \nmid (n-d)$, то

$$\begin{aligned} M(n,d) - i_M(T) &\geqslant M(n,d) - M(n-2,d-1) - M(n-4,d-2) \geqslant \\ &\geqslant (2^{(n-d-1)/2} - 1)(\psi_{d-2} - \psi_{d-4}) + \psi_{d-1} - \psi_{d-3} - 1 > 0. \end{aligned}$$

4. Допустим теперь, что к вершине u примыкает одна цепь на двух вершинах и не примыкает ни одного листа (то есть $\deg u = 2$). Рассмотрим вершину u', соседнюю с u и находящуюся на расстоянии 3 от конца диаметральной цепи P. Возможны четыре случая:
4.1. К вершине u' примыкает хотя бы одна цепь на двух вершинах. Тогда $M(n,d) - i_M(T) \geqslant D_3(n,d)$, где $D_3(n,d) = M(n,d) - M(n-2,d-1) - M(n-3,d-1)$.
4.1.1. Если $n = d+3$, то $D_3(n,d) = 3\psi_{d-2} - \psi_{d-3} - \psi_d > 0$.
4.1.2. Если $n = d+4$, то при $d \geqslant 12$ выполнены соотношения

$$D_3(n,d) = 2\psi_{d-4} - \psi_{d-3} - \psi_{d-5} > 0,$$

а при $8 \leqslant d \leqslant 11$ неравенство $i_M(T) < M(n,d)$ проверяется перебором.
4.1.3. Если $2 \nmid (n-d)$ и $n \geqslant d+5$, то

$$D_3(n,d) \geqslant (2^{(n-d-1)/2} - 1)(\psi_{d-2} - \psi_{d-3}) + \psi_{d-1} - \psi_{d-2} - 1 > 0.$$

4.1.4. Если $2 \mid (n-d)$ и $n \geqslant d+6$, то

$$\begin{aligned} D_3(n,d) &= 2^{(n-d-2)/2}(2\psi_{d-4} - \psi_{d-2}) - \psi_{d-2} + \psi_{d-3} - 1 \geqslant \\ &\geqslant 4(2\psi_{d-4} - \psi_{d-2}) - \psi_{d-2} + \psi_{d-3} - 1 > 0. \end{aligned}$$

4.2. К вершине u' не примыкает ни одной цепи на двух вершинах, но примыкает один лист. В этом случае из утверждения 1 и предположения индукции следует, что $i_M(T) \leqslant 2M(n-3,d-2) < M(n,d)$.
4.3. Степень вершины u' равна 2. В этом случае рассмотрим вершину u'' — соседнюю с u', находящуюся на расстоянии 4 от конца цепи P. Рассмотрим вначале ситуацию, когда к u'' не примыкает ни одной цепи на двух вершинах. Если $d \notin \{9,10\}$, или $2 \nmid (n-d)$, то утверждение теоремы сразу следует из предположения индукции и равенства

$$M(n-2,d-2) + M(n-3,d-3) = M(n,d).$$

Случаи $d = 9$, $2 \nmid n$ и $d = 10$, $2 \mid n$ необходимо рассмотреть отдельно из-за «нестандартного» поведения функции $M(n,d)$ при $d = 7$, $2 \nmid n$:
4.3.1. $d = 9$ и $2 \nmid n$. Если дерево $T \setminus \{w, w'\}$ не изоморфно дереву $\widetilde{B}'_{7,n-2}$ и одновременно дерево $T \setminus \{w, w', u\}$ не изоморфно дереву $\widetilde{B}'_{6,n-3,p}$, то из предположения индукции следует, что

$$i_M(T) \leqslant M(n-2,7) + M(n-3,6) - 2 < M(n,7).$$

Если же дерево $T \setminus \{w, w'\}$ изоморфно дереву $\widetilde{B}'_{7,n-2}$, или дерево $T \setminus \{w, w', u\}$ изоморфно дереву $\widetilde{B}'_{6,n-3,p}$, то дерево T попадает под один из уже разобранных случаев 1, 3, 4.2.
4.3.2 $d = 10$ и $2 \mid n$. Если дерево $T \setminus \{w, w'\}$ не изоморфно дереву $\check{B}'_{d,n-2}$ и одновременно дерево $T \setminus \{w, w', u\}$ не изоморфно дереву $\widetilde{B}'_{7,n-3}$, то из предположения индукции следует, что

$$i_M(T) \leqslant M(n-2,8) + M(n-3,7) - 2 < M(n,7).$$

Если дерево $T \setminus \{w, w'\}$ изоморфно дереву $\check{B}'_{d,n-2}$, то и T имеет вид $\widetilde{B}'_{7,n}$. Если дерево $T \setminus \{w, w', u\}$ изоморфно дереву $\widetilde{B}'_{7,n-3}$, то дерево T попадает под один из уже разобранных случаев 1, 3, 4.2.
4.4. Остается теперь только рассмотреть случай, когда степень вершины u' равна 2, а к вершине u'' примыкает хотя бы одна цепь на двух вершинах. Возможны четыре подслучая:
4.4.1. Дерево T имеет диаметр 8. Тогда $n \geqslant 11$ и T имеет вид $\widehat{B}_{8,n}$ или $\widehat{B}'_{8,n}$. Из приведённой ниже таблицы видно, что в обоих случаях $i_M(T) < M(n,8)$.

T	$i_M(T)$	$M(n,8)-i_M(T)$
$\widehat{B}_{8,n}$	$9\cdot 2^{(n-9)/2}+3$	$2^{(n-9)/2}-1$
$\widehat{B}'_{8,n}$	$9\cdot 2^{(n-10)/2}+4$	$5\cdot 2^{(n-10)/2}-4$

4.4.2. Дерево T имеет диаметр 9. Тогда $n \geqslant 14$ и T имеет вид $\widehat{B}_{9,n,p}$, $\widehat{B}'_{9,n,p}$ или $\widehat{B}''_{9,n,p}$. Из приведённой ниже таблицы видно, что во всех трёх случаях $i_M(T) < M(n,9)$.

T	$i_M(T)$	нижняя оценка для $(M(n,9)-i_M(T))$
$\widehat{B}_{9,n,p}$	$9\cdot 2^{(n-10)/2}+3(2^p+2^{(n-10-2p)/2})+1$	$7\cdot 2^{(n-12)/2}-5$
$\widehat{B}'_{9,n,p}$	$9\cdot 2^{(n-11)/2}+3\cdot 2^p+6\cdot 2^{(n-11-2p)/2}$	$3\cdot 2^{(n-9)/2}-6$
$\widehat{B}''_{9,n,p}$	$9\cdot 2^{(n-12)/2}+6(2^p+2^{(n-12-2p)/2})$	$2^{(n-4)/2}-10$

4.4.3. $d \geqslant 10$ и к вершине u'' в T примыкают t двухвершинных цепей, одна четырёхвершинная цепь, и дерево T', где $\operatorname{diam}(T') \geqslant d-5$. Тогда, если $2 \mid (n-d)$ и $n \geqslant d+6$, то

$$\begin{aligned} M(n,d)-i_M(T) &\geqslant M(n,d)-M(n-2,d)-3\cdot 2^{t-1}\cdot M(n-2t-5,d-4) = \\ &= 2^{(n-d-2)/2}(\psi_{d-1}-3\psi_{d-6})-3\cdot 2^{t-1}(\psi_{d-5}-\psi_{d-6}) \geqslant \\ &\geqslant 2^{(n-d-2)/2}(\psi_{d-1}-3\psi_{d-6})-3\cdot 2^{(n-d-4)/2}(\psi_{d-5}-\psi_{d-6}) = \\ &= 2^{(n-d-4)/2}(2\psi_{d-4}-\psi_{d-3}) > 0. \end{aligned}$$

Если $n = d+4$, то $t = 1$ и

$$\begin{aligned} M(n,d)-i_M(T) &\geqslant M(d+4,d)-M(d+2,d)-3M(d-3,d-4) = \\ &= 4\psi_{d-1}-2\psi_d-2\psi_{d-3} > 0. \end{aligned}$$

Если $2 \nmid (n-d)$, то

$$\begin{aligned} M(n,d)-i_M(T) &\geqslant M(n,d)-M(n-2,d)-3\cdot 2^{t-1}\cdot M(n-2t-5,d-5) = \\ &= 2^{(n-d-1)/2}(\psi_{d-2}-3\psi_{d-7})-3\cdot 2^{t-1}(\psi_{d-6}-\psi_{d-7}) \geqslant \\ &\geqslant 2^{(n-d-1)/2}(\psi_{d-2}-3\psi_{d-7})-3\cdot 2^{(n-d-3)/2}(\psi_{d-6}-\psi_{d-7}) = \\ &= 2^{(n-d-3)/2}(2\psi_{d-2}-3\psi_{d-4}) > 0. \end{aligned}$$

4.4.4. $d \geqslant 10$ и к вершине u'' в T примыкают t двухвершинных цепей, один лист, одна четырёхвершинная цепь, и дерево T', где $\operatorname{diam}(T') \geqslant d-5$. Тогда, если $2 \mid (n-d)$ и $n \geqslant d+6$, то

$$\begin{aligned} M(n,d)-i_M(T) &\geqslant M(n,d)-M(n-2,d)-3\cdot 2^{t-1}\cdot M(n-2t-6,d-5) = \\ &= 2^{(n-d-2)/2}(\psi_{d-1}-3\psi_{d-7})-3\cdot 2^{t-1}(\psi_{d-6}-\psi_{d-7}) \geqslant \\ &\geqslant 2^{(n-d-2)/2}(\psi_{d-1}-3\psi_{d-7})-3\cdot 2^{(n-d-4)/2}(\psi_{d-6}-\psi_{d-7}) = \\ &= 2^{(n-d-4)/2}(2\psi_{d-3}-\psi_{d-4}) > 0. \end{aligned}$$

Если $n = d+4$, то $t = 1$ и

$$\begin{aligned} M(n,d)-i_M(T) &\geqslant M(d+4,d)-M(d+2,d)-3M(d-4,d-5) = \\ &= \psi_{d-1}+2\psi_{d-3}-2\psi_{d-2} > 0. \end{aligned}$$

Если $2 \nmid (n-d)$, то

$$\begin{aligned} M(n,d)-i_M(T) &\geqslant M(n,d)-M(n-2,d)-3\cdot 2^{t-1}\cdot M(n-2t-6,d-4) = \\ &= 2^{(n-d-3)/2}(2\psi_{d-2}-3\psi_{d-6})-3\cdot 2^{t-1}(\psi_{d-5}-\psi_{d-6}) \geqslant \\ &\geqslant 2^{(n-d-3)/2}(2\psi_{d-2}-3\psi_{d-6})-3\cdot 2^{(n-d-5)/2}(\psi_{d-5}-\psi_{d-6}) = \\ &= 2^{(n-d-5)/2}(4\psi_{d-2}-3\psi_{d-3}) > 0. \end{aligned}$$

Теорема доказана. □

Список литературы

[1] А. Б. Дайняк. *О числе независимых множеств в деревьях фиксированного диаметра* // Дискретный анализ и исследование операций, 2009. №2. т. 16. С. 61–73.

[2] A. Frendrup, A. S. Pedersen, A. A. Sapozhenko, P. D. Vestergaard. *Merrifield-Simmons index and minimum Number of Independent Sets in Short Trees* // Department of Mathematical Sciences, Aalborg University. Research Report Series. ISSN 1399-2503. R-2009-03, January 2009. 13pp.

[3] A. S. Pedersen, P. D. Vestergaard. *An upper bound on the number of independent sets in a tree* // Ars Combinatoria. 2007. 84. P. 85–96

[4] H. Prodinger, R. F. Tichy. *Fibonacci numbers of graphs* // Fibonacci Quart., 20(1). pp. 16-–21, 1982.

[5] B. E. Sagan. *A note on independent sets in trees* // SIAM J. Discr. Math. 1988. 1. P. 105–108.

[6] H. S. Wilf *The number of maximal independent sets in a tree* // SIAM J. Alg. Discr. Meth. 1986. 7. P. 125–130.

УДК 681.3.06

СТРУКТУРНЫЙ АНАЛИЗ В ЗАДАЧЕ ДЕКОМПИЛЯЦИИ

© 2009 г. Е. О. Деревенец, К. Н. Долгова

yegord@ispras.ru, katerina@ispras.ru

Кафедра системного программирования

1 Введение

Декомпилятор — это программная система, восстанавливающая программы на языке высокого уровня из программ на языке низкого уровня, из объектного кода или из исполняемых файлов. Декомпиляция требует разработки алгоритмов и методов восстановления информации об исходной программе, которая была утрачена или существенно преобразована в процессе компиляции. Декомпиляция востребована в таких областях информационных технологий, как обеспечение информационной безопасности, поддержка унаследованного кода и т.д.

Обеспечение информационной безопасности требует проверки программы на наличие вредоносного кода, уязвимостей и решения других задач, для которых существуют развитые инструментальные средства, когда доступен исходный код приложений. В частности, для программ на языках высокого уровня, таких как C и C++, существуют развитые инструментальные средства поиска ошибок [1], удобной навигации по исходному коду [2] и другие. Однако на практике часто приходится работать с приложениями, которые доступны только в бинарном виде, и получить исходные коды которых не представляется возможным, во-первых, по причине того, что это приложение было разработано сторонними разработчиками, во-вторых, по причине утраты исходных кодов. Для программ в бинарном представлении инструментальные средства, позволяющие выполнять анализ их работы, развиты хуже, нежели аналогичные инструменты для программ высокого уровня. В частности, для программ на языке ассемблера разрабатывается инструментальное средство [3], которое позволяет анализировать ассемблерную программу на наличие ошибок и уязвимостей, но даже его наличие не обеспечивает возможность решения всех задач анализа низкоуровневых программ с точки зрения информационной безопасности.

Таким образом, разработка инструментального средства, позволяющего восстанавливать бинарную программу или программу на языке ассемблера в программу на языке высокого уровня, актуальна, так как, в частности, наличие такого средства позволит анализировать низкоуровневые приложения, используя развитые методы и инструменты, предназначенные для анализа высокоуровневых приложений.

Декомпиляция может применяться и при анализе программ, исходный код которых доступен. В частности, при разработке высоконадежных приложений в расчёт принимается даже возможность ошибок или некорректных оптимизаций в компиляторе, которые могут привести к неправильной работе программы и появлению уязвимостей [4]. Пример такой оптимизации — устранение компилятором Microsoft Visual C++ .NET вызова функции `memset`. Компилятор может заметить, что после вызова функции `memset` очищенный массив далее нигде не используется, и удалит её вызов [5]. Однако функция `memset` может использоваться для очистки локального буфера, содержащего секретные данные, например пароль, которые должны быть удалены из памяти, чтобы их нельзя было извлечь посредством сохранения содержимого памяти или каким-либо иным способом. Восстановленный декомпилятором код может использоваться для поиска несоответствий между сгенерированным объектным кодом и исходной программой.

Данная работа посвящена одной из подзадач декомпиляции — восстановлению управляющих конструкций. В статье подробно рассмотрены методы восстановления структурных конструкций языка высокого уровня: `if-then`, `if-then-else`, `while`, `do-while`, `switch` и `try-catch` по потоку ассемблерных инструкций. Заметим, что разные высокоуровневые управляющие конструкции программы могут отображаться в одну и ту же последовательность ассемблерных инструкций, например, один и тот же цикл может быть записан с помощью оператора `for`,

`while` или даже посредством комбинирования операторов `if` и `goto`. При декомпиляции требуется восстановить наиболее высокоуровневую управляющую конструкцию из подходящих, в частности, для данного примера наиболее предпочтительным будет восстановление оператора `for`, потом оператора `while`, а восстановление цикла посредством использования операторов `if` и `goto` вообще не желательно.

В статье описан метод восстановления управляющих конструкций, реализованный в декомпиляторе TyDec, который разрабатывается в Институте системного программирования РАН.

Восстановлению стандартных управляющих конструкций `if`, `while`, `for` посвящено достаточное количество работ, и существуют мощные алгоритмы, в то время как восстановлению оператора множественного выбора `switch` в теории и на практике уделено значительно меньше внимания. В статье описан метод восстановления оператора `switch`, который реализован в декомпиляторе TyDec. Также надо отметить, что на практике востребованы методы восстановления программ не только в язык Си, но и в язык Си++, однако наработок как практических, так и теоретических в этой области немного. Множество управляющих конструкций языка Си++ расширяется относительно множества управляющих конструкций языка Си операторами работы с исключительными ситуациями. В статье представлено описание метода, позволяющего восстанавливать операторы работы с исключительными ситуациями языка Си++.

В работе предполагается, что декомпилятору на вход подаётся программа на языке ассемблера для архитектуры IA-32, полученная в результате работы компилятора или восстановленная из объектного кода с помощью дизассемблера. Также предполагается, что исходная программа была написана на языках Си или Си++.

Описываемые в статье алгоритмы и методы с незначительными модификациями могут применяться и для восстановления программ для других платформ и на других языках, допускающих компиляцию в исполняемый код на машинном языке.

Статья имеет следующую структуру. В разделе 2 приводится обзор существующих методов структурного анализа. В разделе 3 представлены методы структурного анализа, реализованные в декомпиляторе TyDec. В разделе 4 подробно отражены особенности реализации обработки исключительных ситуаций в языке C++ и предлагается метод восстановления конструкций возбуждения и обработки исключений.

2 Обзор работ по восстановлению структурных конструкций

Одной из наиболее полных монографий, посвященных разработке трансляторов, является монография [6]. Традиционно компилятор имеет следующую структуру: лексический анализатор, синтаксический анализатор, семантический анализатор, оптимизатор и кодогенератор. Оптимизация и кодогенерация — это наиболее сложные этапы работы компилятора. Одной из подзадач оптимизации является структурный анализ или анализ потока управления.

В монографии [6] рассматриваются два метода анализа потока управления: первый метод основан на построении доминирующих множеств вершин графа потока управления, а второй метод – интервальный анализ, обобщением которого является структурный анализ.

Анализ потока управления, основанный на построении доминирующих множеств вершин графа потока управления, позволяет работать только с циклами и без существенных модификаций неприменим для выявления условных операторов. В этом методе сначала вычисляется доминирующее множество вершин, выполняемое либо с помощью алгоритма последовательной итерации, либо с помощью алгоритма Ленгауэра-Тарьяна. Алгоритм последовательной итерации завершается при достижении доминирующими множествами неподвижной точки. Алгоритм Ленгауэра-Тарьяна работает быстрее, его временная сложность $O(n \cdot \alpha(n))$, однако он более сложный в реализации.

После того как доминирующее множество найдено, выполняется разметка дуг графа потока управления. Дуги графа потока управления классифицируются на прямые, обратные и косые. Прямая дуга — это дуга из доминирующей вершины в доминируемую, обратная дуга — это дуга из доминируемой вершины в доминирующую, все прочие дуги помечаются как косые. Размеченный граф потока управления позволяет выделить управляющие конструкции, например, обратной дуге $m \to n$ соответствует цикл, состоящий из вершины n и всех вершин, из которых доступна m по пути, не содержащему n. В самом общем случае циклу в исходной про-

грамме соответствует компонента сильной связности графа потока управления. Недостатком этого метода является небольшой набор обнаруживаемых структурных конструкций (только циклы) и относительная сложность реализации.

Интервальный анализ основан на последовательном выделении регионов различных типов и объединением их в одну новую вершину графа потока управления. *Регион* — это набор базовых блоков, имеющий не более одной входящей и не более одной исходящей дуги. Простейшим случаем региона является базовый блок. На первой итерации работы алгоритма все базовые блоки помечаются как самостоятельные регионы. Для выделения регионов строится дерево обхода графа потока управления в глубину. Вершины исследуются в обратном порядке обхода. Если два региона соединены только одной дугой, такие регионы объединяются, если вершина является входной точкой циклической или ациклической управляющей конструкции, то регион, соответствующий этой конструкции, выделяется в новую вершину, и соответствующим образом корректируются дуги. Тип конструкции определяется последовательным сравнением подграфов, включающих рассматриваемую вершину, с соответствующими шаблонами. Алгоритм заканчивает работу, когда преобразованный граф не содержит дуг и состоит только из одного региона.

В самой простой версии интервального анализа выполняется выделение только циклов, а условные переходы не анализируются. Обобщением интервального анализа является структурный анализ, где помимо циклов выделяются еще и условные операторы, а именно, выделяются следующие типы регионов: `block`, `if-then`, `if-then-else`, `self loop`, `while loop`, `natural loop`, `improper interval` (неприводимый подграф, строго говоря не являющийся регионом в нашем определении).

Диссертационная работа C. Cifuentes [7] является одной из первых работ, посвященных разработке декомпилятора в язык Си. Метод восстановления структурных конструкций в рассматриваемой работе, реализован автором в декомпиляторе `dcc`. Структурный анализ основан на выделении в исходном графе потока управления управляющих конструкций посредством некоторого преобразования его в семантически эквивалентный граф.

Так как при декомпиляции в общем случае заранее неизвестно, на каком языке была написана исходная программа, автор выделяет множество управляющих конструкций, общих для большинства языков высокого уровня: `composition`, `conditional`, `pre-tested loop`, `single branching conditional`, `n-way conditional`, `post-tested loop`, `endless loop`. В тех случаях, когда исходный граф не может быть структурирован с использованием предопределённого множества структурных конструкций, используется оператор `goto`.

Порядок выделения управляющих конструкций влияет на конечный вид графа. В предлагаемом автором методе используется следующий порядок выделения управляющих конструкций: `n-way conditionals`, `loops`, `2-way conditionals`.

Для поиска циклов каждая вершина x проверяется на наличие входящего в неё обратного ребра из какой-либо вершины y, входящей в тот же регион. Цикл состоит из всех вершин, встречающихся в процессе обхода графа в глубину позже заголовка x и раньше замыкающей вершины y, принадлежащих одному и тому же интервалу и доминируемых заголовком цикла. Циклы, у которых заголовок или замыкающая вершина принадлежат другому циклу, считаются неприводимыми, и при их восстановлении используются операторы `goto`. После выявления циклов определяется их тип. Цикл с предусловием имеет заголовок с двумя исходящими рёбрами и замыкающую вершину с одним обратным ребром в заголовок. Цикл с постусловием, наоборот, имеет замыкающую вершину с двумя рёбрами: в заголовок и вовне цикла. В остальных случаях цикл считается бесконечным.

Условные конструкции и конструкции сокращённого вычисления выражений определяются путём сравнения подграфов с образцами.

На практике встречаются неприводимые графы, содержащие несколько точек входа. Так как алгоритм выделения структурных конструкций не должен изменять семантику графа потока управления, дублирование вершин не применяется, и граф оставляется в исходном виде, а при восстановлении программы используются `goto`.

Изложенные выше методы не полностью решают задачу структурного анализа, так как, с одной стороны, методы, описанные в монографии [6], ориентированы на прямую задачу компиляции. С другой стороны, методы, описанные в работе [7], недостаточно полны, в частности, не поддерживается восстановление оператора множественного выбора `switch` и не поддержи-

вается восстановление обработки исключительных ситуации языка Си++.

3 Структурный анализ в задаче декомпиляции

Структурный анализ, как в контексте прямой задачи — компиляции, так и в контексте обратной задачи основан на анализе графа потока управления. В задаче декомпиляции граф потока управления строится по потоку ассемблерных инструкций.

Каждой подпрограмме, в исходной программе соответствует свой граф потока управления. Отметим, что разбиение графа потока управления на подпрограммы проводится до структурного анализа.

Построение графа потока управления выполняется следующим образом: сначала вся последовательность инструкций разбивается на базовые блоки. В базовые блоки объединяются все инструкции, которые гарантировано выполняются последовательно. Границами базовых блоков являются метки, условные и безусловные переходы, вызовы функций, меняющие поток управления: `exit`, `_exit`, `longjmp`, `__cxa_throw`, `__CxxThrowException` и другие. Построенные таким образом базовые блоки являются вершинами графа потока управления. Далее строятся дуги графа потока управления, соответствующие всем возможным передачам управления между базовыми блоками. Если базовый блок завершается инструкцией перехода, то в граф потока управления добавляется дуга, соединяющая этот базовый блок и базовый блок, в который передается управление. Если базовый блок завершается условным переходом или инструкцией, не меняющей поток управления, то добавляется дуга в следующий базовый блок.

Следует заметить, что не все дуги графа потока управления могут быть построены в процессе статического анализа программы. Например, при косвенных переходах по таблице переходов, как правило, генерируемых при трансляции оператора `switch`, адрес перехода загружается из ячейки памяти. Хотя во многих случаях множество таких адресов переходов может быть построено, в общем случае эта задача алгоритмически неразрешима. Кроме того, в языках, поддерживающих обработку исключительных ситуаций, после инструкции вызова подпрограммы могут выполняться неявные переходы на обработчики исключений. Как следствие, основная сложность восстановления операторов `switch` и `try-catch` заключается в построении дуг графа потока управления, соответствующих неявным переходам.

После того как граф потока управления построен, выполняется восстановление высокоуровневых управляющих конструкций.

Разработанный авторами метод восстановления управляющих конструкций основан на алгоритме, описанном в разделе 2, однако имеет несколько модификаций.

Граф потока управления в процессе анализа перестраивается в граф регионов. Сначала выполняется разметка дуг графа на прямые, обратные и косые. Построение регионов выполняется итеративно. Изначально каждый базовый блок помечается как самостоятельный регион. Далее на граф накладываются шаблоны, соответствующие восстанавливаемым структурным конструкциям: `block`, `if-then`, `if-then-else`, `compound condition`, `endless loop`, `while`, `do-while`, `natural loop`, `switch`. Подграфы, соответствующие этим шаблонам, приведены на рис. 1. Наложение шаблонов представляет собой обход в глубину графа потока управления. Если некоторый путь соответствует шаблону, то все базовые блоки этого пути выделяются в новый регион, после чего наложение выполняется со следующего не пройденного на данной итерации региона. Обход выполняется до тех пор, пока весь граф потока управления не будет представлять собой один регион.

Порядок наложения шаблонов влияет на структуру восстановленной программы. Экспериментально было установлено, что с точки зрения количества восстанавливаемых управляющих конструкций эффективнее сначала выполнять наложение шаблонов, соответствующих циклам, потом шаблонов, соответствующих ациклическим конструкциям.

Признаком цикла является наличие обратной дуги, входящей в регион, с которого начинается обход графа при наложении шаблона. Начиная с региона, в который входит обратная дуга, выполняется продвижение по графу по прямым дугам до тех пор, пока путь не дошел до вершины, из которой начался обход; это обязательно случится, так как в нее входит обратная дуга. Пройденные вершины должны быть восстановлены как цикл. Для определения типа цикла на выделенный подграф накладываются шаблоны циклов.

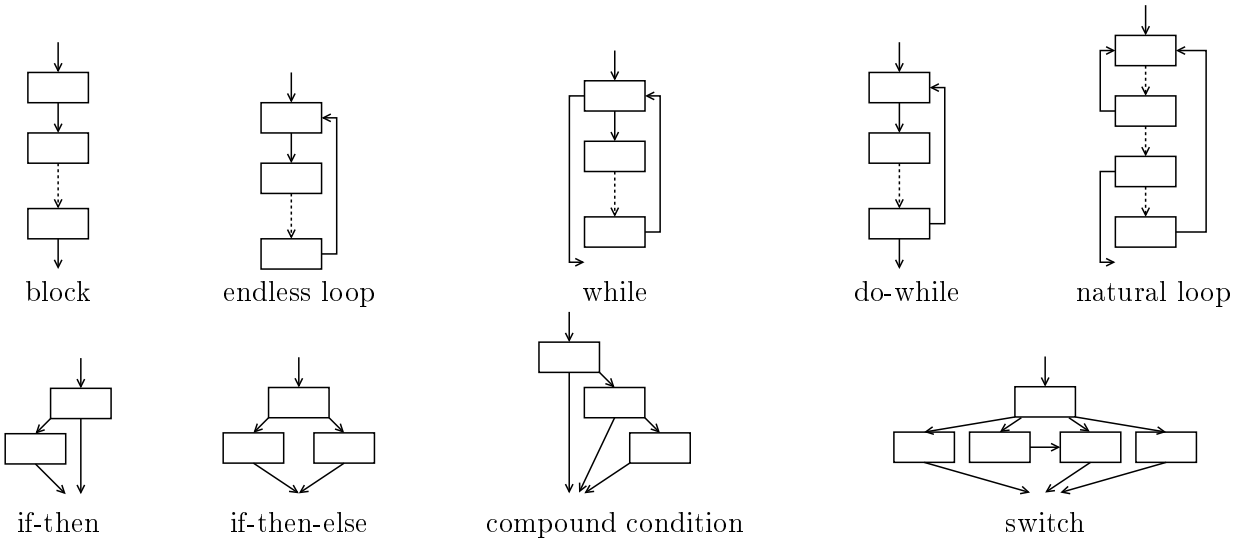


Рис. 1: Типы выделяемых регионов

При восстановлении циклов особого анализа требует восстановление оператора продолжения выполнения `continue` и оператора выхода из середины цикла `break`. Оператор `continue` распознается по наличию дуги из тела цикла в заголовок, а оператор `break` — наличием дуги из тела цикла в выходную вершину цикла. Пусть признаком цикла было наличие обратной дуги из вершины m в вершину n. То есть вершина n доминирует над вершиной m. Тип цикла определяется по следующим признакам:

- если не найдена выходная вершина цикла, то такой цикл считается бесконечным,
- если у вершины n две исходящие дуги, то найден цикл с предусловием,
- если вершины m две исходящие дуги, то найден цикл с постусловием.

После того, как выполнена проверка на наличие циклов, выполняется поиск ациклических управляющих конструкций. Сначала накладывается шаблон, соответствующий оператору множественного выбора `switch`. Потом накладывается шаблон `block`, а затем шаблоны, соответствующие условным переходам, причем сначала выполняется наложение шаблона условного перехода `if-then-else`, а потом — `if-then`.

Восстановление оператора `switch` выполняется с учетом того, что, как правило, он переводится компилятором либо в последовательность сравнений, либо в переход по таблице. Первый способ реализации используется достаточно редко, а реализация через переход по таблице встречается почти всегда, особенно если значения, соответствующие меткам `case`, сгруппированы близко на оси целых чисел.

Первый случай не требует дополнительного рассмотрения, так как при такой реализации оператора `switch` он будет восстановлен в последовательность условных операторов `if`. Во втором случае после вычисления выражения, от которого зависит выбор пути исполнения, выполняется переход по неконстантному выражению. На рис. 2 приведен пример ассемблерного кода, соответствующий оператору выбора `switch` языка Си.

Восстановление оператора `switch` выполняется по нескольким шаблонам. Один из шаблонов включает в себя инструкцию перехода по регистру или обращению к памяти. При нахождении такой инструкции отслеживается значение выражения, по которому происходит переход. Если оно имеет вид `L(,%reg,4)`, где `L` — метка, то она считается указывающей на таблицу переходов для `switch`. В таком случае в граф потока управления добавляются дуги из вершины, содержащей переход по регистру, в `case`-вершины, метки которых записаны в таблице переходов. В процессе структурного анализа вершина-заголовок `switch` и `case`-вершины выделяются в регион типа `switch`.

Пример восстановления декомпилятором оператора `switch` по ассемблерному коду, приведённому на рис. 2, представлен на рис. 3.

```
        cmpl    $6, -8(%ebp)
        ja      L10
        movl    -8(%ebp), %edx
        movl    L11(,%edx,4), %eax
        jmp     *%eax
        .section .rdata,"dr"
        .align 4
L11:
        .long   L3
        .long   L4
        .long   L5
        .long   L6
        .long   L7
        .long   L8
        .long   L9
        .text
```

Рис. 2: Оператор switch, оттранслированный в ассемблер

Представленный метод восстановления структурных конструкций высокого уровня реализован в декомпиляторе *TyDec*, разрабатываемом авторами в Институте системного программирования РАН. Реализация была апробирована на наборе программ с открытым исходным кодом.

В таблице 1 представлены количественные характеристики тестовых примеров. Колонка CLOC содержит количество строк в коде исходной программы. Колонка ALOC содержит количество строк в программе на языке ассемблера, полученной в результате компиляции исходной программы.

Результаты тестирования представлены в таблице 2. Колонки Src содержат количество соответствующих структурных конструкций в исходной программе, Rec — количество конструкций в восстановленной программе. В статистику по восстановлению оператора условного выбора `switch` включены только те операторы, для которых была сгенерирована таблица переходов. Восстановленная программа в тесте `38_deflate.s` содержит больше условных конструкций, чем исходная, из-за того, что цикл `while` был оттранслирован в поток ассемблерных инструкций, соответствующих шаблону оператора `if-do-while`.

Пример	CLOC	ALOC	Описание
`35_wc.s`	241	465	Утилита wc, файл ‘wc.c’
`36_cat.s`	262	618	Утилита cat, файл ‘cat.c’
`37_execute.s`	788	1837	Утилита bc, файл ‘execute.c’
`38_day.s`	503	1383	Утилита calendar, файл ‘day.c’
`39_deflate.s`	763	669	Утилита gzip, файл ‘deflate.c’
`59_lalr.s`	711	1664	Утилита yacc, файл ‘lalr.c’

Таблица 1: Примеры для тестирования

4 Обработка исключительных ситуаций языка Си++

В контексте структурного анализа расширением языка Си++ относительно языка Си является возможность обработки исключительных ситуаций. Исключительные ситуации предназначены для обработки ошибок и являются удобным средством языка при разработке программ на основе независимо спроектированных модулей.

При возникновении исключительной ситуации управление передаётся обработчику исключения. После того, как обработчик исключительной ситуации завершился, возможны два варианта передачи управления, либо на инструкцию программы, в которой возникла исключительная ситуация, либо на инструкцию, следующую за блоком обработки исключительных

Исходная программа

```
switch (getch()) {
    case 'a':
        result = 1;
        break;
    case 'b':
        result = 2;
        break;
    case 'c':
        result = 3;
        break;
    case 'd':
        result = 4;
        break;
    case 'e':
        result = 5;
        break;
    case 'f':
        result = 6;
        break;
    case 'g':
        result = 7;
        break;
    default:
        result = 10;
        break;
}
```

Восстановленная программа

```
eax6 = getch ( );
var8 = eax6 - 97;
if ( ! ( ( unsigned ) var8 > 6 ) ) {
    switch ( var8 ) {
        case 0:
            var9 = 1;
            break;
        case 1:
            var9 = 2;
            break;
        case 2:
            var9 = 3;
            break;
        case 3:
            var9 = 4;
            break;
        case 4:
            var9 = 5;
            break;
        case 5:
            var9 = 6;
            break;
        case 6:
            var9 = 7;
            break;
    }
} else {
    var9 = ( int ) 10;
}
```

Рис. 3: Оператор switch в исходной и восстановленной программах

ситуаций. Обработка по первому сценарию называется обработкой с продолжением, это — устаревший сценарий обработки исключительных ситуаций и в настоящее время практически не поддерживается. Обработка по второму сценарию называется *обработкой без продолжения*, такой сценарий поддерживается всеми современными языками программирования, поддерживающими обработку исключений, и реализован в языке Си++.

Стандарт языка [8] определяет семантику обработки исключительных ситуаций, но не способ её реализации. Формат реализации зависит от конкретного компилятора и платформы, для которой генерируется код.

Далее представлен обзор наиболее распространённых форматов реализации обработки исключительных ситуаций для архитектуры IA-32.

4.1 Формат DWARF2

Формат DWARF2 обработки исключительных ситуаций используется по умолчанию компиляторами GCC и Intel C++ Compiler на GNU/Linux. Общая схема поиска обработчика исключения приведена в [9]. Для раскрутки стека используется Unwind Library [10]. Информация, о стековых фреймах, необходимая для раскрутки стека, содержится в секции `.eh_frame` исполняемого файла, формат которой аналогичен формату отладочной информации DWARF (это и дало название формату) и описан в [11]. Информация об обработчиках хранится в секции `.gcc_except_table`, в виде нескольких областей LSDA (Language Specific Data Area) [9], по одной на каждую функцию. Область данных LSDA представлена на рис. 4. Она содержит информацию о регионах в исполняемом коде функции Sites и соответствующих им обработчиках исключений, информация о которых записана в таблице Action Record Table. Каждому обработчику в Action Record Table поставлен в соответствие указатель на служебную информацию

Пример	ifs		loops		switches	
	Src	Rec	Src	Rec	Src	Rec
35_wc.s	28	19	6	4	1	1
36_cat.s	40	11	6	2	1	1
37_execute.s	71	37	15	9	2	2
38_day.s	42	31	13	7	1	1
39_deflate.s	30	34	15	8	0	0
59_lalr.s	29	26	44	42	0	0
total	240	158	88	72	5	5

Таблица 2: Результаты восстановления структурных конструкций

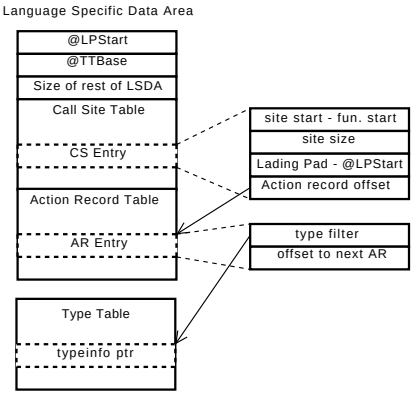


Рис. 4: Структура Language Specific Data Area

`typeinfo` типа обрабатываемого исключения. Указатель на соответствующую область данных LSDA содержится в структурах Frame Description Entry, находящихся в секции `.eh_frame`.

Для возбуждения исключительной ситуации используется функция `__cxa_throw`, принимающая в качестве аргументов указатель на область памяти, которая содержит объект исключения (данный участок памяти выделяется с помощью функции `__cxa_allocate_exception`), указатель на структуру `typeinfo` типа исключения и указатель на деструктор класса исключения (может быть равен нулю).

Главным достоинством формата DWARF2 является отсутствие накладных расходов при входе в `try`-блоки и выходе из них. В качестве недостатка формата DWARF2 можно отметить сложность генерации исключительных ситуаций из обработчиков сигналов и других участков кода, стековый фрейм которых не формируется напрямую компилятором.

Так как вся информация, необходимая для обработки исключительных ситуаций, содержится в статических таблицах известного формата, то восстановление `try-catch` блоков заключается в разборе структур данных в секциях `.eh_frame` и `.gcc_except_table`.

В декомпиляторе *TyDec* реализован метод восстановления обработки исключительных ситуаций в формате DWARF2. Такое расширение декомпилятора позволяет выделять `try`-регионы вместе с соответствующими им `catch`-блоками.

Восстановление конструкции возбуждения исключений `throw` сведено к поиску вызова функции `__cxa_throw` и определению её фактических параметров.

Общая схема работы метода восстановления блоков `try-catch` представлена на рис. 5.

4.2 Формат SjLj

Формат SjLj используется по умолчанию компилятором GCC на платформе Win32. Формат SjLj обработки исключительных ситуаций использует функции `setjmp()` и `longjmp()` или их альтернативные реализации для сохранения и восстановления контекста исполнения.

Во время работы процесса или потока на его стеке поддерживается односвязный список

```
elf = ElfFile(filename)

eh_frame = elf.sections['.eh_frame']
gcc_except_table = elf.sections['.gcc_except_table']

for fde in eh_frame.frame_description_entries:
    print "region start is", fde.pc_begin
    print "region size is", fde.pc_range

    lsda = gcc_except_table.get_lsda_at_offset(fde.lsda)
    for site in lsda.call_sites_table:
        print "try-catch region is at offset", site.offset
        print "try-catch region size is", site.size

        offset = site.action_record_offset - 1

        while offset >= 0:
            action_record = lsda.action_table.get_record_at_offset(offset)
            typeid = lsda.type_table.get_id_at_offset(action_record.type_filter)
            typeinfo = elf.get_type_info(typeid)

            print "this region contains catch-block for type", typeinfo.name

            if action_record.next_offset == 0:
                break
            offset = offset + action_record.next_offset
```

Рис. 5: Алгоритм восстановления блоков try-catch

структур `SjLj_Function_Context`. При входе в функцию, содержащую `try-catch`-блок, на стеке формируется данная структура, содержащая указатель на соответствующую область данных LSDA (Language Specific Data Area), и помещается в конец списка вызовом функции `_Unwind_SjLj_Register`. При выходе из функции последний элемент списка удаляется вызовом функции `_Unwind_SjLj_Unregister`. Определения этих структур и функций можно найти в исходном файле компилятора GCC `unwind-sjlj.c`.

Как и в формате DWARF2, формат SjLj требует размещение области данных LSDA в секции исполняемого файла `.gcc_except_table`. Структура области данных LSDA одинаковая как для формата DWARF2, так и для формата SjLj. Для возбуждения исключительной ситуации используется функция `__cxa_throw`.

Формат SjLj проще, чем формат DWARF2 для реализации, так как в нем решена проблема с генерацией исключений из обработчиков сигналов, однако этот формат требует накладных расходов при входе в функции, содержащие `try`-блоки, даже если код внутри этих блоков никогда не выбрасывает исключений.

Восстановление `try-catch`-блоков в формате SjLj выполнить сложнее, чем в формате DWARF2 из-за необходимости определения значений полей структуры `SjLj_Function_Context`, которая формируется динамически во время работы программы.

4.3 Исключения в компиляторе Microsoft Visual C++

В компиляторе Microsoft Visual C++ исключения реализованы поверх структурной обработки исключений (Structured Exception Handling). Структурная обработка исключений SEH поддерживает во время работы программы стек обработчиков исключений, который реализован в виде односвязного списка структур `EXCEPTION_REGISTRATION` и расположен в стеке процесса или потока [12]. Формат структуры `EXCEPTION_REGISTRATION` представлен на рис. 6. Вершина стека процесса находится по адресу `FS:[0]`. Для реализации исключений языка C++ структура `EXCEPTION_REGISTRATION` расширяется дополнительными полями `id` и `ebp`, расширенная структура представлена на рис. 7.

```
struct EXCEPTION_REGISTRATION
{
    EXCEPTION_REGISTRATION *prev;
    DWORD handler;
    int   id;
    DWORD ebp;
};
```

Рис. 6: Структура EXCEPTION_REGISTRATION

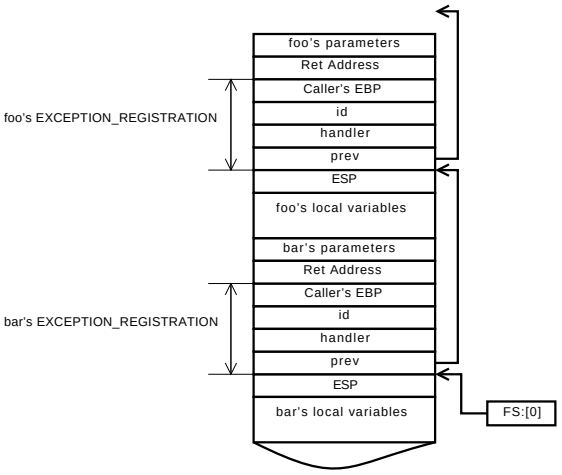


Рис. 7: Структура стека в программе, скомпилированной компилятором MSVC

Обработка одного исключения в формате SEH, как правило, выполняется в два прохода: сначала выполняется поиск подходящего обработчика исключения, начиная с вершины стека, а потом выполняется раскрутка стека до нужного состояния. Компилятор для каждой функции создаёт и регистрирует отдельный обработчик, код которого завершается вызовом функции `__CxxFrameHandler`. Код обработки исключения в формате SEH представлен на рис. 8

```
__ehhandler$?g@@YAXXZ:
        mov     edx, DWORD PTR [esp+8]
        lea     eax, DWORD PTR [edx+12]
        mov     ecx, DWORD PTR [edx-24]
        xor     ecx, eax
        call    @__security_check_cookie@4
        mov     eax, OFFSET __ehfuncinfo$?g@@YAXXZ
        jmp     ___CxxFrameHandler3
```

Рис. 8: Обработка исключения в SEH

Функции `_CxxFrameHandler` через регистр `eax` передаётся указатель на структуру `funcinfo`, содержащую указатель на таблицу `try`-блоков, каждая запись в которой содержит указатель на таблицу соответствующих `catch`-блоков [13]. Поля этой и связанных с ней структур представлены на рис. 9.

Формат обработки исключительных ситуаций, используемый в компиляторе Microsoft Visual C++, имеет наибольшую сложность восстановления , так как используемые при обработке исключений структуры данных формируются динамически в процессе работы программы.

Методы восстановления содержимого этих структур данных является предметом дальнейших исследований.

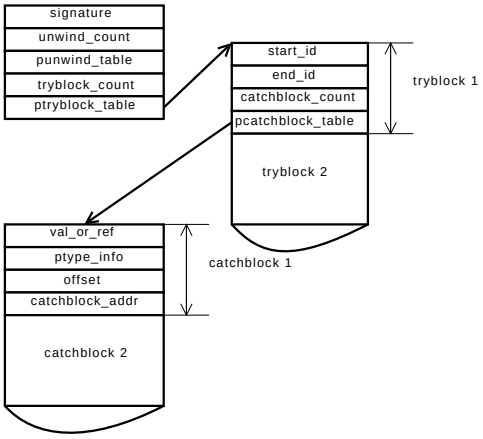


Рис. 9: Структура funcinfo и связанные структуры (основные поля)

5 Заключение

В работе представлен метод восстановления управляющих конструкций языков высокого уровня по ассемблерной программе. Помимо восстановления стандартных управляющих конструкций, таких как циклы и операторы условного перехода, описанный метод позволяет восстанавливать оператор множественного выбора. Представленный метод реализован в разрабатываемом авторами декомпиляторе и апробирован на ряде программ с открытым исходным кодом. Результаты тестирования показали состоятельность представленного подхода.

Помимо этого в работе рассмотрены способы реализации обработки исключений в языке C++ различными компиляторами на архитектуре IA-32. В работе предложен метод восстановления информации об исключениях в формате DWARF2, используемом в ОС Linux и других операционных системах семейства Unix, так как этот формат позволяет полностью восстановить информацию об обработке исключений с помощью только статического анализа исполняемого файла. Представленный метод также реализован в декомпиляторе *TyDec*. Расширение декомпилятора возможностями восстановления структур `try-catch` блоков и типов обрабатываемых исключений позволяет декомпилировать не только программы, которые изначально были написаны на языке Си, но и на Си++. Разработка методов, которые позволят восстанавливать конструкции обработки исключений в формате `SjLj` и в формате, поддерживаемого компилятором MSVC, является направлением дальнейших исследований.

Список литературы

[1] CodeSonar `http://www.grammatech.com/products/codesonar/`
[2] CodeSurfer `http://www.grammatech.com/products/codesurfer/`
[3] G. Balakrishnan and M. Ganai, *PED: Proof-guided Error Diagnosis by Triangulation of Program Error Causes*. Proc. of Software Engineering and Formal Methods (SEFM), November 2008.
[4] G. Balakrishnan, T. Reps, D. Melski, T. Teitelbaum. WYSINWYX: What You See Is Not What You eXecute. 2005.
[5] Michael Howard. Some Bad News and Some Good News. MSDN, October 2002. `http://msdn.microsoft.com/en-us/library/ms972826.aspx`
[6] Steven S. Muchnick. *Advanced Compiler Design and Implementation*, chapter 7. Morgan Kaufmann, 1997.
[7] Cristina Cifuentes. Reverse Compilation Techniques. PhD thesis, Queensland University of Technology, July 1994.
[8] ISO/IEC 14882:1998. Standard for the C++ Programming Language

[9] *Itanium C++ ABI*, Section 7. Exception Handling Tables
http://www.codesourcery.com/public/cxx-abi/exceptions.pdf
[10] *System V Application Binary Interface AMD64 Architecture Processor Supplement*, Section 6.2. Unwind Library Interface, December 2007.
http://www.x86-64.org/documentation/abi.pdf
[11] *Linux Standard Base Core Specification 3.0RC1*, Chapter 8. Exception Frames.
http://refspecs.freestandards.org/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/ehframechpt.html
[12] Matt Pietrek. A Crash Course on the Depths of Win32 Structured Exception Handling. Microsoft Systems Journal, January 1997.
http://www.microsoft.com/msj/0197/exception/exception.aspx
[13] Vishal Kochhar. How a C++ compiler implements exception handling. April 2002.
http://www.codeproject.com/KB/cpp/exceptionhandler.aspx

УДК 519.214.4

ОБ АБСОЛЮТНЫХ КОНСТАНТАХ В РАВНОМЕРНОЙ ОЦЕНКЕ ТОЧНОСТИ НОРМАЛЬНОЙ АППРОКСИМАЦИИ ДЛЯ РАСПРЕДЕЛЕНИЙ, НЕ ИМЕЮЩИХ ТРЕТЬЕГО МОМЕНТА

© 2009 г. М. О. Гапонова, А. Ю. Корчагин, И. Г. Шевцова

margarita.gaponova@gmail.com, ishevtsova@cs.msu.su

Кафедра Математической статистики

1 Постановка задачи и её история

Пусть $X_1, \ldots, X_n$ — независимые одинаково распределённые случайные величины, удовлетворяющие условиям

$$\mathsf{E}X_1 = 0, \quad \mathsf{D}X_1 = 1. \tag{1}$$

Согласно центральной предельной теореме теории вероятностей, в этом случае имеет место равномерная сходимость

$$\rho(F_n, \Phi) \equiv \sup_x |F_n(x) - \Phi(x)| \longrightarrow 0, \quad n \to \infty, \tag{2}$$

где

$$F_n(x) = \mathsf{P}\left(\frac{X_1 + \ldots + X_n}{\sqrt{n}} < x\right), \quad \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt.$$

Указанный факт широко используется при решении задач, возникающих, например, в теории управления запасами, в страховой и финансовой математике, теории измерений, теории надежности, методах Монте–Карло и многих других важных областях. Однако на практике объем выборки всегда конечен и для эффективного применения справедливой при больших n аппроксимации $F_n(x) \approx \Phi(x)$ необходимо иметь разумные оценки скорости сходимости в (2), для конструирования которых условий (1) оказывается недостаточно. А именно, в 1983 году В. К. Мацкявичюс [7] показал, что как бы медленно ни сходилась к нулю последовательность неотрицательных чисел δ_n, найдется последовательность независимых одинаково распределенных случайных величин, удовлетворяющих условиям (1), такая, что для функции распределения F_n соответствующей нормированной суммы при достаточно больших n выполняется неравенство

$$\rho(F_n, \Phi) \geqslant \delta_n.$$

Поэтому в данной работе предполагается ограниченность момента порядка $2 + \delta$ с некоторым $0 < \delta \leqslant 1$:

$$\beta_{2+\delta} \equiv \mathsf{E}|X_1|^{2+\delta} < \infty. \tag{3}$$

Обозначим через $\mathcal{F}_{2+\delta}$ класс всех функций распределения F случайной величины X_1, удовлетворяющих условиям (1) и (3). Тогда для любого $F \in \mathcal{F}_{2+\delta}$ справедливо неравенство (см., например, [10])

$$\rho(F_n, \Phi) \leqslant C(\delta) \cdot L_n^{2+\delta}, \tag{4}$$

где $L_n^{2+\delta}$ — дробь Ляпунова порядка $2 + \delta$:

$$L_n^{2+\delta} = \frac{\beta_{2+\delta}}{n^{\delta/2}},$$

а $C(\delta)$ — некоторая положительная величина, зависящая только от δ.

Случай $\delta = 1$ представляет особый интерес, поскольку, как известно, выполнение условия (3) при $\delta > 1$ не приводит, вообще говоря, к увеличению скорости сходимости в центральной предельной теореме.

Впервые неравенство (4) было доказано при $\delta = 1$ в 1941–1942 гг. независимо друг от друга Э. Берри [20] и К.–Г. Эссеем [21]. История отыскания наименьшего возможного (*правильного*) значения этой постоянной

$$C_* = \sup\left\{\rho(F_n, \Phi)/L_n^3 \colon F \in \mathcal{F}_3,\ n \geqslant 1\right\}$$

довольно интересна и богата результатами. Так, Э. Берри [20] утверждал, что $C_* \leqslant 1.88$, однако, как обнаружил позднее П. Л. Сюй [23], вычисления Берри содержали ошибку. К.–Г. Эссеен [21] показал, что $C_* \leqslant 7.59$. Х. Бергстрём [19] снизил оценку до $C_* \leqslant 4.8$. К. Такано [27] получил результат $C_* \leqslant 2.031$. По-видимому, работа Такано (опубликованная на японском языке) выпала из поля зрения некоторых исследователей, так как в нескольких более поздних публикациях приводятся немного худшие оценки. В частности, в работе Эссеена [22] имеется упоминание о неопубликованных вычислениях, приводящих к неравенству $C_* \leqslant 2.9$. Д. Л. Уоллес [29] получил оценку $C_* \leqslant 2.05$. В. Феллер [9], упоминая результат Уоллеса, также обходит вниманием работу Такано. Вычислению точного значения C_* придавал большое значение А. Н. Колмогоров. В работе [6] он высказал предположение о том, что $C_* = 1/\sqrt{2\pi}$. К сожалению, это предположение оказалось не совсем точным: в 1956 г., решая несколько иную задачу, К.-Г. Эссеен [22] показал, что в (4) постоянная C_* не может быть меньше, чем

$$C_1 = \frac{\sqrt{10}+3}{6\sqrt{2\pi}} = \frac{1}{\sqrt{2\pi}} + 0.0107899...$$

Однако, как позже установил Б. А. Рогозин [11],

$$\limsup_{n\to\infty} \inf_{a,b} \frac{\sigma^3\sqrt{n}}{\beta} \sup_x \left|F_n(x) - \Phi\Big(\frac{x-a}{b}\Big)\right| = \frac{1}{\sqrt{2\pi}} = 0.3989\ldots$$

Тем самым предположение Колмогорова было в определенном смысле подтверждено.

Тем не менее, точное значение C_* абсолютной постоянной в классическом неравенстве Берри–Эссеена до сих пор неизвестно. В 1966-67 гг. В. М. Золотарёв показал, что $C_* \leqslant 1.301$ [4] и $C_* \leqslant 0.82$ [5, 30]. В 1971-72 гг. П. Ван Бик [16, 17] усовершенствовал метод Золотарева, и, используя знакопеременные сглаживающие ядра, получил оценку $C_* \leqslant 0.7882$. Следующий рекорд был установлен И. С. Шигановым [15] десять лет спустя: $C_* \leqslant 0.7655$. При этом в качестве адаптивного ядра использовалась функция более общего вида, с бóльшим количеством параметров. Комбинирование метода Шиганова с асимптотическими оценками Правитца [26], по какой-то причине выпавшими из поля зрения Шиганова, в 2006 г. позволило И. Г. Шевцовой [12] снизить оценку до 0.7056. И, наконец, с помощью еще более глубокой модификации того же метода в 2008 г. была получена наилучшая на сегодняшний день оценка $C_* \leqslant 0.7005$ [13].

В 1989 году В. Бенткус и К. Кирша [18] показали, что неравенство (4) для $\delta = 1$ и $n = 1$ выполняется с $C(1) \leqslant 0.37$.

В 2006 году С. В. Нагаев и В. И. Чеботарев [8] доказали неравенство

$$\rho(F_n, \Phi) \leqslant 0.515489 \cdot L_n^3 + 0.427675 \cdot \frac{1}{\sqrt{n}} + 0.153597 \cdot \frac{1}{n}, \quad n \geqslant 1,$$

из которого получили некоторые “условные” верхние оценки $C(1)$:

	$n \geqslant 3$	$n \geqslant 10$
$\beta_3 \geqslant 3$	$C(1) \leqslant 0.687607$	$C(1) \leqslant 0.674238$
$\beta_3 \geqslant 4$	$C(1) \leqslant 0.644578$	$C(1) \leqslant 0.634551$
$\beta_3 \geqslant 10$	$C(1) \leqslant 0.567124$	$C(1) \leqslant 0.563114$

Случай $0 < \delta < 1$ изучен гораздо меньше. В 1983 году В. Тысиак [28] (также см. работу Г. Падитца [25]) получил следующие оценки константы: $C(\delta) \leqslant C_T(\delta)$, где

δ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$C_T(\delta)$	1.102	1.076	1.008	0.950	0.902	0.863	0.833	0.812	0.802

В 1986 году Г. Падитц [24] показал, что при $\delta = 0$ имеет место неравенство

$$\rho(F_n, \Phi) \leqslant 3.51 \cdot \mathsf{E}\left(X_1^2 \min\left\{1, \frac{|X_1|}{\sqrt{n}}\right\}\right),$$

откуда вытекает равномерная по $\delta \in [0, 1)$ оценка $C(\delta) \leqslant 3.51$, так как при любом $\delta \in (0, 1]$ выражение в правой части последнего неравенства не превосходит $3.51 \cdot L_n^{2+\delta}$.

Совсем недавно в работе авторов [2] были получены следующие *асимптотические* оценки константы $C(\delta)$ при $L_n^{2+\delta} \leqslant 0.4$:

δ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$C(\delta) \leqslant$	0.6995	0.9230	0.8527	0.7722	0.7049	0.6674	0.6518	0.6508	0.6611

существенно уточняющие результат Тысиака при $L_n^{2+\delta} \leqslant 0.4$. В данной работе, комбинируя метод Тысиака с результатом статьи [2], мы уточняем *абсолютные* оценки Тысиака и показываем, что для всех $L_n^{2+\delta}$ и $n \geqslant 1$ справедливы оценки $C(\delta) \leqslant \overline{C}(\delta)$, где

δ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\overline{C}(\delta)$	1.0739	1.0001	0.9407	0.8876	0.8454	0.8126	0.7876	0.7720	0.7671

В следующем разделе изложены основные идеи доказательства этого результата.

2 Основные идеи доказательства

В основе доказательства лежит неравенство сглаживания Золотарёва [3], улучшенное Ван Биком [16, 17] (и использованное Тысиаком в [28]), а также вышеупомянутый результат работы [2]. Ниже мы опишем лишь ключевые этапы, формулируя соответствующие утверждения в виде лемм.

Пусть $p(x)$ — некоторая абсолютно интегрируемая функция. Обозначим

$$\hat{p}(t) = \int_{-\infty}^{\infty} e^{itx} p(x) dx, \quad \|p\| = \int_{-\infty}^{\infty} |p(x)| dx,$$

$$p^+(x) = \max\{p(x), 0\}, \quad p^-(x) = -\min\{p(x), 0\}.$$

Для произвольных $x, y > 0$ введём функции

$$V(x) = x \int_0^x p^+(u) du, \quad q(y) = \frac{1}{\sqrt{2\pi}} \int_0^{\infty} |\hat{p}(t/y)| \frac{r(t)}{t} dt, \quad r(t) = |f_n(t) - e^{-t^2/2}|,$$

где $f_n(t)$ – характеристическая функция, соответствующая функции распределения $F_n(x)$. Обозначим через Λ класс всех непрерывных симметричных интегрируемых функций p, имеющих интегрируемое преобразование Фурье $\hat{p}$ и таких, что

$$\int_0^{\infty} p^+(t)\, dt > \int_0^{\infty} p^-(t)\, dt.$$

Лемма 1 (см. [16]). *Для любого* $p \in \Lambda$

$$\rho(F_n, \Phi) \leqslant \sqrt{\frac{2}{\pi}} \cdot \frac{V(x)/y + q(y)}{4V(x) - x\,\|p\|}, \quad x > \chi_p,\ y > 0,$$

где χ_p *— единственный положительный корень уравнения*

$$4V(x) - x\,\|p\| = 0.$$

Замечание 1. Поскольку правая часть последнего неравенства инвариантна относительно умножения ядра на положительное число, можно ограничиться классом ядер $p \in \Lambda$, удовлетворяющих условию нормировки $\|p\| = 1$.

Следующая лемма, доказанная В. Тысиаком в диссертации [28], дает оценку величины $r(t)$, фигурирующей в $q(y)$, равномерную в классе всех распределений, удовлетворяющих условиям (1) и (3).

Обозначим

$$\varepsilon = L_n^{2+\delta} = \frac{\beta_{2+\delta}}{n^{\delta/2}}.$$

Лемма 2. 1°. *Для* $|t| < \sqrt{2}\varepsilon^{-1/(2+\delta)}$

$$r(t) \leqslant r_1(t) = e^{-t^2/2}\left(\exp\left\{L_\delta\left(|t|\varepsilon^{1/(2+\delta)}\right)\right\} - 1\right),$$

где

$$L_\delta(x) = \gamma(\delta)x^{2+\delta} - \frac{x^2}{2} - \ln\left(1 - \frac{x^2}{2}\right), \quad \gamma(\delta) = \sup_{x\in\mathbb{R}} \frac{|e^{ix} - 1 - ix + x^2/2|}{|x|^{2+\delta}}.$$

2°. *Для всех* $t \in \mathbb{R}$

$$r(t) \leqslant r_2(t) = e^{-t^2/2}\left(\exp\left\{2\varkappa(\delta)\varepsilon|t|^{2+\delta}\right\} + 1\right),$$

где

$$\varkappa(\delta) = \sup_{x\in\mathbb{R}} \frac{|\cos x - 1 + x^2/2|}{|x|^{2+\delta}}.$$

3°. *Для всех* $t \in \mathbb{R}$

$$r(t) \leqslant r_3(t) = 1 + e^{-t^2/2}.$$

В качестве мажоранты $|f_n(t) - e^{-t^2/2}|$ естественно выбирать минимум из r_1, r_2, r_3. Положим

$$r_0(t,\varepsilon) = \min\{r_1(t), r_2(t), r_3(t)\}, \quad q(y,\varepsilon) = \frac{1}{\sqrt{2\pi}}\int_0^\infty |\hat{p}(t/y)|\frac{r_0(t,\varepsilon)}{t}\,dt.$$

$$D(\varepsilon, x, y, p) \equiv \sqrt{\frac{2}{\pi}} \cdot \frac{V(x)/y + q(y,\varepsilon)}{\varepsilon(4V(x) - x\,\|p\|)}.$$

Вышеприведенные леммы позволяют искать оценку $\overline{C}(\delta)$ константы $C(\delta)$ в виде

$$\overline{C}(\delta) = \sup_\varepsilon D(\varepsilon), \quad D(\varepsilon) \equiv \inf\{D(\varepsilon, x, y, p)\colon x > \chi_p,\ y > 0,\ p \in \Lambda\}.$$

Следующие две леммы играют важную роль при вычислении $\sup_\varepsilon D(\varepsilon)$.

Лемма 3 (см. [28]). *Для любых* $0 < \varepsilon_1 \leqslant \varepsilon \leqslant \varepsilon_2$ *справедливо соотношение*

$$D(\varepsilon) \leqslant D(\varepsilon_2) \cdot \frac{\varepsilon_2}{\varepsilon_1}.$$

Лемма 4 (см. [1]). *Для любой функции распределения $F(x)$ с нулевым средним и единичной дисперсией справедлива оценка*

$$\sup_x |F(x) - \Phi(x)| \leqslant \sup_{b>0} \left(\Phi(b) - \frac{b^2}{1+b^2} \right) < 0.540937.$$

Результаты работы [2] и леммы 4 позволяют ограничиться рассмотрением $0.4 < \varepsilon < 0.55/\overline{C}(\delta)$ и искать $\overline{C}(\delta)$ в виде

$$\overline{C}(\delta) = \sup_{0.4<\varepsilon<0.55/\overline{C}(\delta)} D(\varepsilon).$$

Последовательность функций $p_m \in \Lambda$, на которой достигается инфимум по $p \in \Lambda$ функционала

$$D^*(\varepsilon, p) = \inf\{D(\varepsilon, x, y, p)\colon x > \chi_p,\ y > 0\} = \inf_{\substack{x>\chi_p,\\ y>0}} \left\{ \sqrt{\frac{2}{\pi}} \cdot \frac{V(x)/y + q(y,\varepsilon)}{\varepsilon(4V(x) - x\,\|p\|)} \right\}$$

построена Ван Биком [17]. Обозначим

$$\operatorname{supp} \hat{p} = \{t\colon \hat{p}(t) > 0\},$$

Λ_1 — класс функций $p \in \Lambda$ с компактным носителем $\operatorname{supp}\hat{p}$, Λ_0 — класс функций $p \in \Lambda_1$, для которых отрезок $[-1, 1]$ является минимальным отрезком, содержащим $\operatorname{supp}\hat{p}$.

Лемма 5 (см. [17]). *Для любых фиксированных $n \geqslant 1$ и $\varepsilon > 0$*

$$\inf_{p\in\Lambda} D^*(\varepsilon, p) = \inf_{p\in\Lambda_0} D^*(\varepsilon, p) = \inf_{p\in\Lambda_1} D^*(\varepsilon, p).$$

Пусть $h(t)$ — произвольная функция из Λ_1, для которой $\hat{h}(t)$ не обращается в нуль на отрезке $[-1, 1]$. Обозначим

$$p_m(x, \mathbf{a}, \mathbf{w}) = \frac{1}{2} \sum_{i=1}^{m} w_i (h(x + a_i) + h(x - a_i)), \tag{5}$$

где $\mathbf{a} = (a_1, \ldots, a_m)$, $\mathbf{w} = (w_1, \ldots, w_m)$, $a_i,\ w_i \in \mathbb{R}$, $i = \overline{1, m}$, такие, что $p_m \in \Lambda$. Заметим, что

$$\hat{p}_m(t, \mathbf{a}, \mathbf{w}) = \hat{h}(t) \sum_{i=1}^{m} w_i \cos(a_i t).$$

Лемма 6 (см. [17]). *Для определенной выше последовательности функций $\{p_m\}_{m\geqslant 1}$ имеем*

$$\inf_{p\in\Lambda} D^*(\varepsilon, p) = \lim_{m\to\infty} \inf_{\mathbf{a},\mathbf{w}} D^*(\varepsilon, p_m), \tag{6}$$

где инфимум берется по всем $\mathbf{a} = (a_1, \ldots, a_m)$, $\mathbf{w} = (w_1, \ldots, w_m)$ и $a_i,\ w_i \in \mathbb{R}$, $i = \overline{1, m}$, таким, что $p_m \in \Lambda$.

Таким образом, последние два утверждения позволяют заменить непараметрическую задачу минимизации функционала $D(\varepsilon, p)$ на параметрическую задачу минимизации функции $D(\varepsilon, p_m)$ по переменным $a_i,\ w_i,\ i = \overline{1, m}$ с достаточно большим m. Однако, слишком большие значения m снижают эффективность работы численных методов многомерной оптимизации, поэтому необходим некоторый компромисс: нужно использовать максимальное m, при котором эффективность работы численных методов еще достаточно велика. Так, например, для некоторых функций $h(t) \in \Lambda_1$ в данной работе была проведена численная оптимизация $D(\varepsilon, p_m)$ с

p_m, определенным в (5). Текущая реализация программы, производящей необходимые вычисления, позволила провести расчеты для $m \leqslant 7$. В качестве функций $h(x)$ выбиралась одна из следующих трех:

$$B(x) = \frac{1-\cos x}{\pi x^2}, \qquad \hat{B}(t) = (1-|t|)^+,$$
$$S(x) = \frac{\sin x}{2\pi x(1-x^2/\pi^2)}, \qquad \hat{S}(t) = \cos^2\frac{\pi t}{2}\mathbf{1}(|t| \leqslant 1),$$
$$J(x) = \frac{15J_{3/2}^2\left(\frac{x}{2}\right)}{|x|^3}, \qquad \hat{J}(t) = (1-5t^2+5|t|^3-|t|^5)^+,$$

где $J_{3/2}(x)$ — функция Бесселя порядка 3/2. Функция $B(x)$ использовалась еще Берри, $S(x)$ и $J(x)$ предложены Ван Биком.

Численная оптимизация проводилась по методу сопряжённых градиентов. При написании соответствующей программы использовалась библиотека математических функций GNU Scientific Library (GSL) на языке C (http://www.gnu.org/software/gsl/). Максимум функции $D(\varepsilon)$ достигался примерно в точке $\varepsilon = 0.4-0.6$ в зависимости от δ: с ростом δ точка максимума сдвигается вправо, функция $D(\varepsilon)$ как бы "размазывается", а ее максимум становится менее выраженным и сдвигается вправо.

3 Результаты вычислений

Для $\delta = 1$ были получены следующие результаты (данная таблица включает в себя также результаты, полученные в работе [14]):

$h(x)=S(x)$		$h(x)=B(x)$		$h(x)=J(x)$	
m	$\overline{C}(1)$	m	$\overline{C}(1)$	m	$\overline{C}(1)$
2	0.7056	3	0.7664	2	0.7032
3	0.7047	4	0.7658	3	0.7021
4	0.7029	5	0.7281	4	0.7011
5	0.7008			5	0.7009
6	0.7005			6	0.7008
				7	0.7006

Уже при $m \geqslant 4$ наблюдается некоторая стабилизация. Данное наблюдение позволает выдвинуть гипотезу, что предельное значение (6) близко к 0.7005.

Несмотря на теоретическое отсутствие зависимости предела (6) от вида конкретной функции h, ключевым фактором успеха при практическом расчете, когда значение m фиксировано, является удачный выбор некоторой комбинации компонент ядра:

$$p(x) = \frac{1}{2}\sum_{j=1}^{k}\sum_{i=1}^{m_j} w_{i,j}(h_j(x+a_{i,j}) + h_j(x-a_{i,j})), \quad \hat{p}(t) = \sum_{j=1}^{k} h_j(t)\sum_{i=1}^{m_j} w_{i,j}\cos(a_{i,j}t), \qquad (7)$$

где $k \geqslant 1$ — число различных *типов* порождающих ядер, $m_j \geqslant 1$ — количество компонент порождающего ядра j-го типа, $h_j(t)$ — произвольные функции из Λ_1, $a_{i,j}, w_{i,j}$, $i = \overline{1,m_j}$, $j = \overline{1,k}$ — произвольные вещественные числа, обеспечивающие выполнение условия $p \in \Lambda$. При этом общее число компонент ядра равняется $m_1 + \ldots + m_k \equiv m$, а число параметров — $2m$ (отметим также, что число *свободных* параметров при этом равняется $2m-1$, поскольку адаптивное ядро из неравенства сглаживания Золотарева инвариантно относительно умножения на положительное число, и один из весов можно зафиксировать). Ниже приведена таблица полученных результатов для $\delta = 1$ и $k = 2$.

$h_1(x)=S(x),\ h_2(x)=B(x)$				$h_1(x)=S(x),\ h_2(x)=J(x)$			
m_1	m_2	m	$\overline{C}(1)$	m_1	m_2	m	$\overline{C}(1)$
1	1	2	0.7244	2	3	5	0.7009
2	2	4	0.7066	3	2	5	0.7015
4	2	6	0.7020	3	3	6	0.7005
5	1	6	0.7018	4	1	5	0.7015
				5	1	6	0.7006
				5	2	7	0.7005
				2	5	7	0.7006

Приведенные в последних двух таблицах результаты согласуются с полученными ранее [13]. Наилучшими для $\delta = 1$ оказались 6-компонентные смеси $S(x)$, 7-компонентные смеси $J(x)$, а также смеси, состоящие из 5 компонент $S(x)$ и 1 или 2 компонент $J(x)$, и 7-компонентная смесь, содержащая 2 компоненты $S(x)$ и 5 компонент $J(x)$. Наиболее простые из этих ядер были выбраны для проведения вычислений при $0 < \delta < 1$, а именно, мы рассмотрели 6-компонентные смеси $S(x)$, а также $6-7$-компонентные смеси $J(x)$, то есть ядра вида (5), в которых $m = 6$, $h(x) = S(x)$; $m = 6$, $h(x) = J(x)$; $m = 7$, $h(x) = J(x)$. Полученные результаты содержатся в следующей таблице.

$m=6,\ h(x)=S(x)$		$m=6,\ h(x)=J(x)$		$m=7,\ h(x)=J(x)$	
δ	$\overline{C}(\delta)$	δ	$\overline{C}(\delta)$	δ	$\overline{C}(\delta)$
0.1	1.0754	0.1	1.0741	0.1	1.0739
0.2	1.0032	0.2	1.0008	0.2	1.0001
0.3	0.9438	0.3	0.9407	0.3	0.9407
0.4	0.8922	0.4	0.8878	0.4	0.8876
0.5	0.8479	0.5	0.8466	0.5	0.8454
0.6	0.8149	0.6	0.8126	0.6	0.8126
0.7	0.7941	0.7	0.7876	0.7	0.7876
0.8	0.7728	0.8	0.7720	0.8	0.7720
0.9	0.7671	0.9	0.7672	0.9	0.7671

Из этой таблицы видно, что 7-компонентная смесь $J(x)$ дает наилучшие оценки. Ниже приводится сравнительная таблица наших результатов с результатами Тысиака. В первом столбце указаны значения δ, во втором — оценки Тысиака: $C(\delta) \leqslant C_T(\delta)$, в третьем — оценки, полученные в данной работе: $C(\delta) \leqslant \overline{C}(\delta)$, в четвертом — разность $C_T(\delta) - \overline{C}(\delta)$.

δ	$C_T(\delta)$	$\overline{C}(\delta)$	$C_T(\delta) - \overline{C}(\delta)$
0.1	1.102	1.074	0.028
0.2	1.076	1.001	0.075
0.3	1.008	0.941	0.067
0.4	0.95	0.888	0.062
0.5	0.902	0.846	0.056
0.6	0.863	0.813	0.050
0.7	0.833	0.788	0.045
0.8	0.812	0.772	0.040
0.9	0.802	0.768	0.030

Расчеты проводились на вычислительном комплексе “Regatta” (http://regatta.cs.msu.su), установленном на факультете ВМК МГУ им. М. В. Ломоносова[1] с конфигурацией: 16 Power4 процессоров, 64 Gb оперативной памяти. Процесс вычисления $\overline{C}(\delta)$ для десяти значений δ проводился на 10 независимых друг от друга потоках и в среднем занимал около 7 часов для каждого ядра.

[1]Авторы выражают благодарность Д. А. Гуляеву за предоставленные вычислительные ресурсы.

В заключение авторы выражают искреннюю признательность В. Ю. Королёву за постоянное внимание к работе.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проекты 08-01-00563, 08-01-00567 и 08-07-00152, Совета по грантам президента Российской Федерации для поддержки молодых российских ученых – кандидатов наук №МК-654.2008.1, а также грантом У.М.Н.И.К.

Список литературы

[1] Р. Н. Бхаттачария и Р. Ранга Рао. *Аппроксимация нормальным распределением.* — М., "Наука", 1982.

[2] М. О. Гапонова, И. Г. Шевцова. Асимптотические оценки абсолютной постоянной в неравенстве Берри–Эссеена для распределений, не имеющих третьего момента. — *Информатика и ее применения*, 2009, в печати.

[3] В. М. Золотарёв. О близости распределений двух сумм независимых случайных величин. — *Теория вероятн. и ее примен.*, 1965, т. 10, вып. 3, с. 519-526.

[4] В. М. Золотарёв. Абсолютная оценка остаточного члена в центральной предельной теореме. — *Теория вероятн. и ее примен.*, 1966, т. 11, вып. 1, с. 108-119.

[5] В. М. Золотарев. Некоторые неравенства теории вероятностей и их применение к уточнению теоремы А. М. Ляпунова. — *ДАН СССР*, 1967, т. 177, №3, с. 501-504.

[6] А. Н. Колмогоров. Некоторые работы последних лет в области предельных теорем теории вероятностей. – *Вестник Моск. ун-та*, 1953, №10, с. 29-38.

[7] В. К. Мацкявичюс. О нижней оценке скорости сходимости в центральной предельной теореме. — *Теория вероятн. и ее примен.*, 1983, т. 28, вып. 3, с. 565-569.

[8] С. В. Нагаев, В. И. Чеботарев. Новый подход к оценке абсолютной константы в неравенстве Берри—Эссеена. — *Тез. докл. XXXI Дальневосточной школы-семинара им. акад. Е. В. Золотова*, Владивосток, 2006, с. 19.

[9] В. Феллер. *Введение в теорию вероятностей и ее приложения. Т. 2.* — М., "Мир", 1984.

[10] В. В. Петров. *Суммы независимых случайных величин.* Москва, "Наука", 1972.

[11] Б. А. Рогозин. Одно замечание к работе Эссеена "Моментное неравенство с применением к центральной предельной теореме". — *Теория вероятн. и ее примен.*, 1960, т. 5, вып. 1, с. 125-128.

[12] И. Г. Шевцова. Уточнение верхней оценки абсолютной постоянной в неравенстве Берри–Эссеена. — *Теория вероятн. и ее примен.*, 2006, т. 51, вып. 3, с. 622-626.

[13] И. Г. Шевцова. Уточнение абсолютной константы в классическом неравенстве Берри-Эссеена. — *Статистические методы оценивания и проверки гипотез*, изд-во Пермского государственного университета, Пермь, 2008, т. 21, с. 159-168

[14] И. Г. Шевцова. Об абсолютной постоянной в неравенстве Берри-Эссеена. — *Сборник статей молодых ученых факультета ВМК*, Макс–Пресс, Москва, 2008, т. 5.

[15] И. С. Шиганов. Об уточнении верхней константы в остаточном члене центральной предельной теоремы. – *Проблемы устойчивости стохастических моделей. Труды ВНИИСИ*, 1982, с. 109-115.

[16] P. van Beek. Fourier-analytische Methoden zur Verscharfung der Berry–Esseen Schranke. – *Doctoral dissertation*, Friedrich Wilhelms Universitat, Bonn, 1971.

[17] P. van Beek. An application of Fourier methods to the problem of sharpening the Berry–Esseen inequality. – *Z. Wahrsch. verw. Geb.*, 1972, Bd. 23, p. 187-196.

[18] V. Bentkus, K. Kirsha. Estimates of closeness of a distribution function to the normal law. — *Lithuanian math. J.*, 1989, vol. 29, p. 321-332.

[19] H. Bergström. On the central limit theorem in the case of not equally distributed random variables. – *Skand. Aktuarietidskr.*, 1949, vol. 33, p. 37-62.

[20] A. C. Berry. The accuracy of the Gaussian approximation to the sum of independent variates. — *Trans. Amer. Math. Soc.*, 1941, vol. 49, p. 122–139.

[21] C.–G. Esseen. On the Liapunoff limit of error in the theory of probability. — *Ark. Mat. Astron. Fys.*, 1942, vol. A28, No. 9, p. 1–19.

[22] C.-G. Esseen. A moment inequality with an application to the central limit theorem. – *Skand. Aktuarietidskr.*, 1956, vol. 39, p. 160-170.

[23] P. L. Hsu. The approximate distributions of the mean and variance of a sample of independent variables. – *Ann. Math. Statist.*, 1945, Vol. 16, No. 1, p. 1-29.

[24] H. Paditz. Über eine Fehlerabschätzung im zentralen Grenzwertsatz. — *Wiss. Z. Hochschule für Verkehswesen "Friedrich List"*. Dresden. 1986, B. 33, H. 2, S. 399-404.

[25] H. Paditz. On the error-bound in the nonuniform version of Esseen's inequality in the L_p-metric. — *Statistics*, 1996, vol. 27, p. 379-394.

[26] H. Prawitz. On the remainder in the central limit theorem. — *Scand. Actuarial J.*, 1975, No. 3, pp. 145-156.

[27] K. Takano. A remark to a result of A. C. Berry. – *Res. Mem. Inst. Math.*, 1951, vol. 9, No. 6, p. 4.08-4.15.

[28] W. Tysiak. *Gleichmäßige und nicht-gleichmäßige Berry–Esseen–Abschätzungen*. Dissertation, Wuppertal, 1983.

[29] D. L. Wallace. Asymptotic approximations to distributions. – *Ann. Math. Statist.*, 1958, Vol. 29, p. 635-654.

[30] V. M. Zolotarev. A sharpening of the inequality of Berry–Esseen. – *Z. Wahrsch. verw. Geb.*, 1967, Bd. 8, p. 332-342.

УДК 512.57

О ПОЛОЖЕНИИ САМОДВОЙСТВЕННЫХ k-ЗНАЧНЫХ ФУНКЦИЙ В РЕШЁТКЕ ЗАМКНУТЫХ КЛАССОВ

© 2009 г. В. Б. Ларионов

VitalyBLarionov@yandex.ru

Кафедра Математической кибернетики

1 Основные понятия.

Обозначим множество $E_k = \{0, 1, \ldots, k-1\}$.

Определение 1. Функция $f(x_1, \ldots, x_n)$ называется функцией k-значной логики ($k \geqslant 2$), если она определена на E_k^n и все её значения принадлежат E_k.

Будем использовать следующие стандартные обозначения. Множество всех функций k-значной логики обозначим P_k. Для любого подмножества A из P_k через $[A]$ будем обозначать замыкание относительно операции суперпозиции (для функций далее везде будет идти речь именно об этом типе замыкания).

Пусть на E_k задана некоторая подстановка σ.

Определение 2. Функция k-значной логики $f(x_1, \ldots, x_n)$ называется самодвойственной относительно подстановки σ, если выполнено следующее тождество: $f(x_1, \ldots, x_n) = \sigma^{-1}(f(\sigma(x_1), \ldots, \sigma(x_n)))$, где через σ^{-1} мы обозначаем подстановку, обратную к σ. Известно ([3]), что множество всех функций из P_k, самодвойственных относительно σ, является замкнутым классом. Обозначим этот класс через S_σ.

Определение 3. Пусть $p(x_1, \ldots, x_m)$ – некоторый предикат, определённый на E_k^m, $f(y_1, \ldots, y_n)$ – функция из P_k. Говорят, что функция $f(y_1, \ldots, y_n)$ сохраняет предикат $p(x_1, \ldots, x_m)$, если для любых n наборов $\tilde{a_i} = (a_{i1}, \ldots, a_{im})$ $(i = 1, \ldots, n)$, удовлетворяющих предикату p, набор $f(a_{11}, \ldots, a_{n1}), \ldots, f(a_{1m}, \ldots, a_{nm})$ также удовлетворяет предикату p. По определению будем считать, что тождественно ложный предикат сохраняет любая функция.

Будем обозначать через $\mathrm{Pol}(p)$ множество функций, сохраняющих предикат p.

Лемма 1. $S_\sigma = \mathrm{Pol}(R_\sigma)$, где R_σ – двухместный предикат, которому удовлетворяют все наборы вида $(i, \sigma(i))$ и только они.

Доказательство

1. Предположим, что $f \in S_\sigma$. Возьмём в таблице для f две строки, удовлетворяющие предикату R_σ: $\tilde{a} = (a_1, \ldots, a_n)$ и $\sigma(\tilde{a}) = (\sigma(a_1), \ldots, \sigma(a_n))$. Получаем, что $f(\tilde{a}) = \sigma^{-1}(f(\sigma(\tilde{a})))$, то есть пара значений на этих строках также удовлетворяет предикату R_σ. Отсюда следует, что f сохраняет предикат R_σ и $f \in \mathrm{Pol}(R_\sigma)$.

2. Пусть теперь $f \in \mathrm{Pol}(R_\sigma)$. Следовательно, для любой пары строк в таблице $\tilde{a}$ и $\tilde{b}$, удовлетворяющей покомпонентно предикату R_σ, выполнено $f(\tilde{b}) = \sigma(f(\tilde{a}))$. Получаем, что $f \in S_\sigma$ по определению.

Одним из шести семейств предполных классов ([2]) является семейство, обозначаемое **S** — подмножество множества всех самодвойственных функций. Известно ([3]), что самодвойственный класс является предполным (принадлежит множеству **S**) тогда и только тогда, когда подстановка, сохраняемая им, распадается в произведение циклов одинаковой простой длины. В связи с этим возникает вопрос о том, какое положение в решётке замкнутых классов занимают остальные самодвойственные классы.

Отметим, что в работе [6] показано, что над произвольным самодвойствееным классом находится конечное число замкнутых классов.

Нам в дальнейшем понадобится семейство **C**.

Определение 4. Класс функций A принадлежит семейству **C**, если A — множество функций, сохраняющих центральный предикат p, т.е. предикат, обладающий следующими свойствами:

1. Абсолютная симметричность. Для любой подстановки σ на множестве $\{1,\dots,n\}$ и любого набора $\tilde{a} \in E_k^n$ справедливо $p(a_1,\dots,a_n) = TRUE \iff p(a_{\sigma(1)},\dots,a_{\sigma(n)}) = TRUE$.

2. Абсолютная рефлексивность. Предикат p истинен на любом наборе $\tilde{a}$, таком что найдутся номера $1 \leqslant i,j \leqslant m$ (m — местность p), удовлетворяющие $a_i = a_j$ (в $\tilde{a}$ присутствуют хотя бы две равные компоненты).

3. Существует центр предиката p, то есть существует элемент $h \in E_k$ такой, что для любого набора $\tilde{a} \in E_k^n$, у которого есть компонента, равная h, справедливо $p(\tilde{a}) = TRUE$.

Далее мы приведём некоторые результаты о соответствии Галуа между замкнутыми классами предикатов и функций.

Пусть T – некоторая система предикатов.

Определение 5.([5])

1. Если p – обозначение m-местного предиката из T, $x_1,\dots,x_m$ – некоторые различные символы переменных, то $p(x_1,\dots,x_m)$ – формула над T, реализующая предикат $p(x_1,\dots,x_m)$. В этой формуле все переменные $x_1,\dots,x_m$ – свободные, связанные переменные отсутствуют.

2. Пусть $A(y_1,\dots,y_l)$ – формула над T со свободными переменными $y_1,\dots,y_l$, реализующая предикат $p(y_1,\dots,y_l)$. Тогда $(\exists y_i)A(y_1,\dots,y_i,\dots,y_l)$ – формула над T со свободными переменными $y_1,\dots,y_{i-1},y_{i+1},\dots,y_l$ и связанными переменными такими же, как у $A(y_1,\dots,y_l)$, с добавлением y_i, реализующая предикат $\exists y_i p(y_1,\dots,y_i,\dots,y_l)$.

3. Пусть опять $A(y_1,\dots,y_l)$ – формула над T со свободными переменными $y_1,\dots,y_l$, реализующая предикат $p(y_1,\dots,y_l)$, $x_{i_1},\dots,x_{i_l}$ – не обязательно различные символы переменных, отличающиеся от символов связанных переменных формулы $A(y_1,\dots,y_l)$. Тогда $A(x_{i_1},\dots,x_{i_l})$ – формула над T со свободными переменными $x_{i_1},\dots,x_{i_l}$ (тут имеются в виду все различные переменные из списка $x_{i_1},\dots,x_{i_l}$) и связанными переменными такими же, как у $A(y_1,\dots,y_l)$, реализующая предикат $p(x_{i_1},\dots,x_{i_l})$.

4. Пусть $A(x_1,\dots,x_l)$ и $B(y_1,\dots,y_m)$ – формулы над T с множествами свободных переменных соответственно $x_1,\dots,x_l$ и $y_1,\dots,y_m$, реализующие предикаты $p_1(x_1,\dots,x_l)$ и $p_2(y_1,\dots,y_m)$, причём множества всех переменных этих формул (свободных и связанных) не пересекаются. Тогда $A(x_1,\dots,x_l)\&B(y_1,\dots,y_m)$ – формула над T со свободными переменными $x_1,\dots,x_l,y_1,\dots,y_m$, реализующая предикат $p_1(x_1,\dots,x_l)\&p_2(y_1,\dots,y_m)$. Множество связанных переменных этой формулы – объединение множеств связанных переменных формул $A(x_1,\dots,x_l)$ и $B(y_1,\dots,y_m)$.

Замыканием $[T]$ системы T называется множество предикатов, реализуемых всевозможными формулами над T.

Далее для простоты изложения будем называть операции над предикатами следующим образом: добавление квантора существования (пункт 2 определения) – проекцией по соответствующей переменной, подстановку вместо $y_{j_1},\dots,y_{j_h}$ одного и того же символа переменной (пункт 3 определения) – отождествлением переменных $y_{j_1},\dots,y_{j_h}$, операцию, описанную в пункте 4 – произведением предикатов p_1 и p_2.

Определение 6. Предикаты R_1 и R_2 называются эквивалентными, если $\mathrm{Pol}(R_1) = \mathrm{Pol}(R_2)$. Под $R_1 = R_2$ будем подразумевать равенство предикатов как функций.

Отметим, что если $R_1 \in [R_2]$, то $\mathrm{Pol}(R_2) \subseteq \mathrm{Pol}(R_1)$ ([1]). Этот факт будем обозначать $(*)$.

Обозначим через $\mathrm{Inv}(f)$ множество всех предикатов, каждый из которых сохраняется функцией f. Для произвольных множества функций A и множества предикатов T положим $\mathrm{Inv}(A) = \cap_{f\in A}\mathrm{Inv}(f)$, $\mathrm{Pol}(T) = \cap_{p\in T}\mathrm{Pol}(p)$.

Основными результатами теории Галуа для замкнутых классов являются соотношения ([4]):

$$A = \mathrm{Pol}(\mathrm{Inv}(A)), \quad T = \mathrm{Inv}(\mathrm{Pol}(T)), \tag{1}$$

где A – замкнутый класс функций, содержащий все селекторные функции (т.е. функции вида $f(x_1,\dots,x_n) = x_i$), а T – замкнутый класс предикатов, содержащий все диагональные отношения (т.е. отношения вида $p(x_1,\dots,x_n) \equiv (x_{i_1} = x_{j_1})\&\cdots\&(x_{i_l} = x_{j_l})$).

Предположим, что подстановка σ разлагается в произведение k_i циклов длины l_i ($i = 1, \ldots, s$). Очевидно, что $\sum_{i=1}^{s} k_i l_i = k$.

Нетрудно заметить, что существует минимальное число h такое, что $\sigma^h = e$, где e — тождественная подстановка.

Лемма 2. $h = \mathrm{HOK}(l_1, \ldots, l_s)$.

На протяжении всей работы мы будем обозначать через h указанную величину для подстановки σ.

Далее рассмотрим некоторые замкнутые классы, содержащие класс S_σ.

2 Классы семейства S_σ.

Рассмотрим всевозможные классы вида S_{σ^i}, $2 \leqslant i \leqslant h$.

Лемма 3. $S_\sigma \subseteq S_{\sigma^i}$ для любого $i > 1$.

Доказательство Из леммы 1 следует, что $S_\sigma = \mathrm{Pol}(R_\sigma)$ и $S_{\sigma^i} = \mathrm{Pol}(R_{\sigma^i})$. По определению предиката R_σ имеем: $R_{\sigma^i}(x_1, x_2) = \exists y_1, \ldots, y_{i-1} R_\sigma(x_1, y_1) \& R_\sigma(y_1, y_2) \& \ldots \& R_\sigma(y_{i-1}, x_2)$. В силу $(*)$ получаем утверждение леммы.

Напомним, что h — минимальное число, удовлетворяющее $\sigma^h = e$.

Следствие. Если $c = ab \pmod h$, то $S_{\sigma^a} \subseteq S_{\sigma^c}$.

Доказательство Обозначим подстановку σ^a через π. По лемме 3 получаем, что $S_\pi \subseteq S_{\pi^b} = S_{\sigma^{ab}} = S_{\sigma^c}$.

Лемма 4. Пусть $1 < i < h$, $i_1 = \mathrm{HOD}(i, h)$. Тогда $S_{\sigma^i} = S_{\sigma^{i_1}}$.

Доказательство Пусть $i = i_1 i_2$. По условию леммы числа h и i_2 взаимно просты. Используя алгоритм Евклида ([7]), получаем, что существуют целые числа a, b такие, что справедливо: $ah + bi_2 = 1$, что эквивалентно $bi_2 = 1 \pmod h$. Домножая последнее равенство на i_1, имеем $bi = i_1 \pmod h$. По следствию леммы 3 получаем $S_{\sigma^i} \subseteq S_{\sigma^{i_1}}$. С другой стороны, непосредственно из условия данной леммы следует, что $S_{\sigma^{i_1}} \subseteq S_{\sigma^i}$.

Лемма доказана.

Тот факт, что число a делится на число b, будем обозначать через $b|a$.

Следствие. Любой класс вида S_{σ^i}, где i не делит h, совпадает с некоторым классом S_{σ^a}, где $a|h$.

Изучим вопрос о том, как устроена подстановка σ^a.

Рассмотрим произвольный цикл подстановки σ. Не ограничивая общности рассуждения, предположим, что он состоит из элементов $0, \ldots, l-1$ и $\sigma(i) = i + 1 \pmod l$ на элементах указанного цикла.

Лемма 5. Элементы $0, \ldots, l-1$ рассматриваемого цикла образуют d циклов равной длины $\frac{l}{d}$ в подстановке σ^a, где $d = \mathrm{HOD}(a, l)$.

Доказательство Возьмём произвольный элемент b рассматриваемого цикла подстановки σ. В подстановке $\pi = \sigma^a$ в один цикл с элементом b попадут все элементы, имеющие вид $\pi^i(b) = \sigma^{ia}(b) = b + ia \pmod l$. Найдётся такое минимальное число s, что $\pi^s(b) = b$ или $b + sa = b \pmod l$.

Последнее условие эквивалентно $sa = 0 \pmod l$ или

$$sa = lt. \tag{2}$$

Число s в данном выражении является длиной рассматриваемого цикла подстановки π.

Обозначим в уравнении (2) $r = sa$. Поскольку, как мы оговорили выше, s — минимальное число, удовлетворяющее (2), то $r = \mathrm{HOK}(a, l)$. Длина рассматриваемого цикла

$$s = \frac{r}{a} = \frac{\mathrm{HOK}(a, l)}{a}.$$

В случае $s < l$ указанный цикл распадается на

$$\frac{l}{s} = \frac{al}{\mathrm{HOK}(a, l)} = \mathrm{HOD}(a, l)$$

циклов равной длины, в каждый из которых входят элементы, имеющие один и тот же остаток от деления на a.

Лемма доказана.

Пусть число h представимо в виде $h = h_1^{\alpha_1} \dots h_r^{\alpha_r}$, где $h_1, \dots, h_r$ — простые числа.

Лемма 6. Пусть a и b — различные делители числа h. $S_{\sigma^a} \subset S_{\sigma^b}$ тогда и только тогда, когда $a|b$.

Доказательство Пусть число a не делит b. Представим указанные числа в виде:

$$a = h_1^{a_1} \dots h_r^{a_r},$$

$$b = h_1^{b_1} \dots h_r^{b_r},$$

где $0 \leqslant a_i, b_i \leqslant \alpha_i$ и существует номер i_0 такой, что $a_{i_0} > b_{i_0}$.

Напомним, что мы предполагаем, что в подстановке σ имеется k_i циклов длины l_i ($i = 1, \dots, s$). Заметим, что для любого j существует цикл длины l_i такой, что $h_j^{\alpha_j} | l_i$. В самом деле, если для некоторого j не существует указанного цикла, то мы приходим к противоречию с определением числа h.

Итак, существует l_{j_0} такое, что $h_{i_0}^{\alpha_{i_0}} | l_{j_0}$. Следовательно, в наибольший общий делитель a и l_{j_0} входит сомножитель $h_{i_0}^{a_{i_0}}$, а в наибольший общий делитель b и l_{j_0} — не входит. Обозначим $d_a = \mathrm{HOD}(a, l_{j_0})$, $d_b = \mathrm{HOD}(b, l_{j_0})$. Из сказанного выше следует, что $d_a > 1$, d_a не делит d_b.

Итак, по лемме 5 получаем, что цикл длины l_{j_0} подстановки σ при возведении в степень a распадётся на d_a циклов, а при возведении в степень b – на d_b циклов.

Покажем, что при этом $S_{\sigma^a} \not\subseteq S_{\sigma^b}$.

В самом деле, пусть цикл из элементов $\{0, \dots, l_{j_0} - 1\}$ при возведении в степень a распадается на циклы $\Delta_1^1, \dots, \Delta_{d_a}^1$, а при возведении в степень b – на циклы $\Delta_1^2, \dots, \Delta_{d_b}^2$ (буквами мы здесь обозначили множества элементов, из которых состоит каждый цикл). Определим функцию одной переменной $f(x)$ так, чтобы она каждое множество Δ_j^1 переводила в себя и действовала на Δ_j^1 по следующему правилу: $f(a) = \sigma^{s_j}(a)$. Числа s_j свои для каждого множества, мы их пока не фиксируем. На остальных наборах положим $f(x) = x$.

Покажем, что $f \in S_{\sigma^a}$ для любых чисел s_j. Возьмём произвольные элементы $t_1, t_2 \in E_k$, принадлежащие указанному циклу (в ином случае утверждение очевидно) и удовлетворяющие $R_{\sigma^a}(t_1, t_2) = TRUE$. Понятно, что при этом t_1, t_2 принадлежат одному и тому же множеству Δ_j^1. Тогда

$$R_{\sigma^a}(f(t_1), f(t_2)) = R_{\sigma^a}(f(t_1), f(\sigma^a(t_1))) = R_{\sigma^a}(\sigma^{s_j}(t_1), \sigma^{a+s_j}(t_1)) =$$
$$= R_{\sigma^a}(t_3, \sigma^a(t_3)) = TRUE,$$

где $t_3 = \sigma^{s_j}(t_1)$.

Итак, множества $\Delta_1^1, \dots, \Delta_{d_a}^1, \Delta_1^2, \dots, \Delta_{d_b}^2$ – два разных разбиения множества $\{0, \dots, l_{j_0} - 1\}$. Если число d_a не делит число d_b, то второе разбиение не может быть измельчением первого. В этом случае найдутся два элемента k_1 и $k_2 = \sigma^b(k_1)$ (то есть соседние в некотором Δ_l^2), принадлежащие разным множествам Δ_i^1 и Δ_j^1. Пусть $f(k_1) = a_1$ и $f(k_2) = a_2$.

Пусть вначале множества Δ_i^1 или Δ_j^1 содержат хотя бы по два элемента. Покажем, что мы можем выбрать числа s_i и s_j так, чтобы $R_{\sigma^b}(a_1, a_2) = FALSE$. Положим $s_j = 0$, откуда $f(k_2) = k_2$, а s_i выберем равным a (в этом случае $a_1 = \sigma^a(k_1)$ будет отличным от k_1 элементом множества Δ_i^1). Поскольку $R_{\sigma^b}(k_1, k_2) = TRUE$, то $R_{\sigma^b}(a_1, k_2) = FALSE$ по определению предиката, задающего подстановку. Получаем, что $f \notin S_{\sigma^b}$.

Если же все множества Δ_i^1 одноэлементные, то в качестве $f(x)$ достаточно взять функцию, принимающую на множестве $0, \dots, l_{j_0} - 1$ значение 1, на остальных наборах $f(x) = x$. Очевидно, что $f \in S_{\sigma^a}$. Поскольку множества Δ_j^2 не могут быть одноэлементными, то $f \notin S_{\sigma^b}$.

Докажем теперь обратное утверждение. Пусть $a|b$. Поскольку числа a и b различны, то b не делит a. По уже доказанному получаем, что $S_{\sigma^b} \not\subseteq S_{\sigma^a}$. С другой стороны, по следствию леммы 3 имеем $S_{\sigma^a} \subseteq S_{\sigma^b}$. Из двух указаннных соотношений получаем требуемое:

$$S_{\sigma^a} \subset S_{\sigma^b}.$$

Лемма доказана.

Из последней леммы, а также следствия к лемме 4 получаем следующий результат.

Теорема 1. Решётка классов S_{σ^i} изоморфна решётке делителей числа h.

Обозначим через $\mathbf{S}_\sigma$ множество классов S_{σ^i}, где i — делитель числа h, $i \neq 1$, $i \neq h$.

Приведём некоторые свойства описанной решётки классов $\mathbf{S}_\sigma$.

1. Число классов: $\prod_{i=1}^{r} (\alpha_i + 1) - 2$.

2. Высота: $\sum_{i=1}^{r} \alpha_i - 1$.

3. Ширина верхнего и нижнего слоёв: r.

3 Классы семейства $\mathbf{T}_\sigma$.

Опять будем предполагать, что в подстановке σ имеется k_i циклов длины l_i (i=1,. . . ,s). Обозначим через $M_{d,\sigma}$ подмножество множества E_k, состоящее из всех элементов, образующих циклы подстановки σ, длины которых делят число d.

Определение 7. Пусть подстановка π определена на подмножестве $M_{d,\sigma}$ множества E_k для некоторого d и равна σ на указанном множестве. Классом $H_{d,\pi}$ назовём множество функций k-значной логики, сохраняющих следующий предикат: $R_\pi(x_1, x_2) = TRUE$ тогда и только тогда, когда $x_1, x_2 \in M_{d,\pi}$ и $x_2 = \pi(x_1)$.

Из определения множества $M_{d,\sigma}$ следует, что, если мы возьмём число d_1, не делящее число h, то найдётся число d_2, являющееся делителем h, такое, что $M_{d_1,\sigma} = M_{d_2,\sigma}$. Поэтому мы можем в дальнейшем рассматривать в качестве чисел d только делители h.

Обозначим через $\mathbf{T}_\sigma$ множество классов H_{d,π^t}, где d — делитель числа h, π — подстановка, определённая на множестве $M_{d,\sigma}$ ($M_{d,\sigma} \neq E_k$, $M_{d,\sigma} \neq \emptyset$) и совпадающая с подстановкой σ на нём, t — делитель числа q, являющегося минимальным, удовлетворяющим $\pi^q = e$. Ограничения на параметр t следуют из того факта, что для фиксированного множества $M_{d,\sigma}$ (и, соответственно, подстановки π) для структуры классов H_{d,π^t} справедливы результаты предыдущего раздела с заменой E_k на $M_{d,\sigma}$. Иными словами, решётка классов H_{d,π^t} над $H_{d,\pi}$ изоморфна решётке делителей числа q.

Лемма 7. Любой класс из семейства $\mathbf{T}_\sigma$ содержит класс S_σ.

Доказательство Рассмотрим предикат $R_{\pi^t}(x_1, x_2)$, задающий класс H_{d,π^t}. Нетрудно убедиться в выполнении следующего соотношения:

$$R_{\pi^t}(x_1, x_2) = \exists y_1, \ldots, y_{t+d-2} R_\sigma(x_1, y_1) \& R_\sigma(y_1, y_2) \& \ldots \& R_\sigma(y_{t-1}, x_2) \&$$

$$\& R_\sigma(x_1, y_t) \& R_\sigma(y_t, y_{t+1}) \& \ldots \& R_\sigma(y_{t+d-2}, x_1).$$

То есть $R_{\pi^t} \in [R_\sigma]$. Используя $(*)$, получаем требуемое.

Лемма 8. Пусть d_1, d_2 — различные делители h такие, что в подстановке σ существует цикл длины l, которая делит число d_1 и не делит d_2, t — любое число. Введём для указанных чисел подстановки π_1 и π_2, как описано выше. Тогда

1. Классы H_{d_1,π_1^t} и H_{d_2,π_2^t} различны.

2. $H_{d_1,\pi_1^t} \subset H_{d_2,\pi_2^t}$ тогда и только тогда, когда $d_2 | d_1$.

Доказательство Классы H_{d_1,π_1^t} и H_{d_2,π_2^t} задают соответственно предикаты $R_{\pi_1^t}$ и $R_{\pi_2^t}$. Обозначим через C указанный в лемме цикл подстановки σ. Он входит в область определения подстановки π_1 и не входит в область определения подстановки π_2. Рассмотрим функцию одной переменной:

$$f(x) = \begin{cases} 0, & x \in C, \\ x, & x \in E_k \setminus C. \end{cases}$$

Очевидно, $f(x) \in H_{d_2,\pi_2^t}$. Однако, на цикле C предикат $R_{\pi_1^t}$, состоящий из пар $(i, \pi_1^t(i))$ на столбце значений будет ложен, и $f(x) \notin H_{d_1,\pi_1^t}$.

Если числа d_1 и d_2 не делят друг друга, то данный пример можно использовать в обе стороны. При этом мы получим $H_{d_1,\pi_1^t} \nsubseteq H_{d_2,\pi_2^t}$, $H_{d_2,\pi_2^t} \nsubseteq H_{d_1,\pi_1^t}$.

Если же $d_2|d_1$, то $R_{\pi_2^t} \in [R_{\pi_1^t}]$, поскольку

$$R_{\pi_2^t}(x_1, x_2) = \exists y_1, \ldots, y_{d_2} R_{\pi_1^t}(x_1, x_2) \& R_{\pi_1^t}(x_1, y_1) \& R_{\pi_1^t}(y_1, y_2) \& \ldots \& R_{\pi_1^t}(y_{d_2-1}, x_1).$$

В силу $(*)$ и уже доказанного первого пункта леммы получаем $H_{d_1,\pi_1^t} \subset H_{d_2,\pi_2^t}$.

Лемма доказана.

4 Надструктура классов S_σ.

Пусть предикат $p \in [R_\sigma]$, где R_σ – предикат, задающий самодвойственный класс S_σ. Запишем p в виде формулы F. Заметим, что в формуле F можно вынести вперёд все кванторы существования, не изменив предикат p. Везде далее мы будем считать, что предикат задаётся формулой именно такого вида.

Сопоставим F ориентированный граф G_F по следующему правилу: между множеством вершин G_F и множеством переменных F (учитываем и свободные и связанные) существует взаимно однозначное соответствие. Вершину, соответствующую переменной x, будем обозначать v_x. В графе G_F есть ориентированное ребро (v_x, v_y) тогда и только тогда, когда в формуле F содержится запись $R(y, x)$.

Пусть $v_{y_1}, \ldots, v_{y_N}$ – все вершины графа G_F, первые n из которых соответствуют свободным переменным, остальные — связанным.

Определение 8. Путём из вершины v_{y_1} в вершину v_{y_m} в графе G_F будем называть любую последовательность рёбер вида $\{(v_{y_1}, v_{y_2}), (v_{y_2}, v_{y_3}), \ldots, (v_{y_{m-1}}, v_{y_m})\}$ (вершины и рёбра могут повторяться). Ориентация рёбр указанной последовательности может быть любая. Циклом называется путь, в котором первая и последняя вершины совпадают. Длиной пути будем называть разность количества рёбер, пройденных в прямом направлении, и количества ребёр, пройденных в обратном.

Определение 9. Матрицей расстояний для графа G_F будем называть матрицу D_F размера h на h, элементами которой являются множества d_{ij}, построенные по следующему правилу: число $0 \leqslant l < h$ принадлежит d_{ij} тогда и только тогда, когда в графе G_F существует путь из вершины v_{y_i} в вершину v_{y_j} длины l' и $l' = l \pmod h$, других элементов в d_{ij} нет.

Далее рассмотрим случай, когда G_F связен.

Если граф G_F не содержит циклов или содержит только циклы длины, кратной h, то каждое множество d_{ij} состоит из единственного элемента.

Лемма 9. Пусть граф G_F содержит циклы с ненулевыми длинами $c_1, \ldots, c_q$. Тогда он содержит циклы длины $\sum_{i=1}^q \alpha_i c_i$ для любых целых $\alpha_1, \ldots, \alpha_q$.

Доказательство Мы можем пройти цикл длины c_1 α_1 раз (знак числа α_1 указывает направление обхода). Поскольку граф связный, существует путь S' от какой-нибудь вершины указанного цикла к какой-то вершине цикла длины c_2. Пройдём по нему в прямом направлении, пройдём цикл длины c_2 α_2 раз, затем вернёмся по S' к первому циклу. Получим цикл длины $\alpha_1 c_1 + \alpha_2 c_2$ (поскольку длина пути S' из-за его прохождения в прямом и обратном направлении будет присутствовать в сумме с плюсом и с минусом). Рассуждая аналогично для циклов с длинами $c_3, \ldots, c_q$, получим требуемое.

Лемма 10. Граф G_F содержит циклы с ненулевыми длинами $c_1, \ldots, c_q$ тогда и только тогда, когда он содержит цикл длины $c = \mathrm{HOD}(|c_1|, \ldots, |c_q|)$.

Доказательство Пусть граф G_F содержит циклы с длинами $c_1, \ldots, c_q$. Если среди указанных длин есть отрицательные, сделаем их положительными, изменив направления обхода соответствующих циклов. Согласно алгоритму Евклида ([7]), существуют целые числа $b_1, \ldots, b_q$ такие, что $\sum_{i=1}^q b_i c_i = \mathrm{HOD}(c_1, \ldots, c_q) = c$. Используя лемму 9, получаем, что граф G_F содержит цикл длины c.

С другой стороны, пусть G_F содержит цикл длины $c = \mathrm{HOD}(c_1, \ldots, c_q)$. Пусть $c_i = \beta_i c$, $(i = 1, \ldots, q)$. Тогда, обходя указанный цикл $\beta_1, \ldots, \beta_q$ раз, получим, что в графе G_F есть циклы с длинами $c_1, \ldots, c_q$ (опять знаки чисел β_i указывают направления обхода).

Лемма доказана.

Следствие. Каждый элемент d_{ij} матрицы D_F содержит только числа $(s_{ij} + \alpha c)$ $(\mod h)$ для всех α, где $0 \leqslant s_{ij} < c$ (s_{ij} может быть нулевым).

Доказательство Пусть минимальным неотрицательным элементом d_{ij} является s_{ij}. По леммам 9 и 10 множество d_{ij} содержит все элементы $(s_{ij} + \alpha c)$ $(\mod h)$ для любых α. Следовательно выполнено $0 \leqslant s_{ij} < c$.

Предположим теперь, что d_{ij} содержит некоторый элемент s'_{ij} отличный от указанного множества, то есть не существует числа α такого, что $s'_{ij} = s_{ij} + \alpha c$. Получаем, что между вершинами v_{y_i} и v_{y_j} в графе G_F существует два пути с различными длинами s_{ij} и s'_{ij}. Следовательно, в графе существует цикл длины $s'_{ij} - s_{ij} \neq 0$. По определению числа c справедливо $c|(s'_{ij} - s_{ij})$, или существует β такое, что $\beta c = s'_{ij} - s_{ij}$, откуда $s'_{ij} = s_{ij} + \beta c$. Полученное противоречие доказывает следствие.

Напомним: мы условились считать, что вершины $v_{y_1}, \ldots, v_{y_n}$ и только они соответствуют свободным переменным формулы F; через $M_{d,\sigma}$ мы обозначили подмножество множества E_k, состоящее из всех элементов, образующих циклы подстановки σ, длины которых делят число d.

Положим M_p равным E_k, если граф G_F не содержит циклов или длины всех его циклов кратны h, или $M_{c,\sigma}$ в противном случае. Оставим в каждом множестве d_{ij} по одному минимальному неотрицательному элементу s_{ij}.

Лемма 11. Предикату p удовлетворяют следующие наборы и только они:

$$\begin{vmatrix} x_1 & x_2 & x_3 & \ldots & x_i & \ldots & x_n \\ \sigma^{s_{i1}}(a_j) & \sigma^{s_{i2}}(a_j) & \sigma^{s_{i3}}(a_j) & \ldots & a_j & \ldots & \sigma^{s_{in}}(a_j), \end{vmatrix}$$

где a_j — элементы множества M_p, s_{iq} — указанные выше числа, i — произвольное, но фиксированное число, удовлетворяющее неравенству $1 \leqslant i \leqslant n$.

Доказательство Покажем вначале, что никакие другие наборы не удовлетворяют предикату. Зафиксируем любое число $1 \leqslant i \leqslant n$.

Пусть $M_{c,\sigma} \neq \emptyset$. Возьмём произвольный элемент $b \in E_k \setminus M_{c,\sigma}$. Предположим, что $p(a_1, \ldots, a_n) = TRUE$, где $a_i = b$. По лемме 10 существуют циклы, проходящие через вершину v_{y_i}, с длинами s_{ii} и $s_{ii} + c$. Следовательно, существует подобный цикл длины c. По определению графа G_F получаем, что $\sigma^c(b) = b$. Это означает, что элемент b входит в цикл подстановки σ длины, делящей число c. Откуда $b \in M_{c,\sigma}$ — противоречие. Таким образом, все компоненты удовлетворяющих предикату p наборов лежат во множестве $M_{c,\sigma}$.

Пусть опять $p(a_1, \ldots, a_n) = TRUE$. По определению матрицы D_F между вершинами v_{y_i} и v_{y_j} существует путь длины s_{ij}. Следовательно, должно выполняться соотношение $a_j = \sigma^{s_{ij}}(a_i)$.

Итак, мы показали, что никакие наборы, кроме тех, что указаны в формулировке леммы, не могут удовлетворять предикату p. Покажем теперь, что остальные наборы (указанные в лемме) удовлетворяют предикату.

Пусть набор $\tilde{a}$ удовлетворяет формулировке леммы. Присвоим каждой переменной y_j предиката p (свободной и связной) значение, равное $\sigma^{s_{ij}}(a_i)$. Покажем далее, что $p(\tilde{a}) = TRUE$, то есть указанное присвоение непротиворечиво. Рассмотрим две произвольные вершины v_{y_q} и v_{y_j} графа G_F. Эти переменные примут значения $a_q = \sigma^{s_{iq}}(a_i)$ и $a_j = \sigma^{s_{ij}}(a_i)$, откуда

$$a_q = \sigma^{s_{iq} - s_{ij}}(a_j). \tag{3}$$

Соединив пути от вершины v_{y_j} до вершины v_{y_i} длины s_{ji} и от вершины v_{y_i} до вершины v_{y_q} длины s_{iq}, мы получим путь от вершины v_{y_j} до вершины v_{y_q} длины $s_{ji} + s_{iq}$. По следствию из леммы 10 между рассматриваемыми вершинами v_{y_j} и v_{y_q} в графе G_F существуют только пути с длинами $s_{jq} + \alpha c$, следовательно, существует число β_1 такое, что $s_{ji} + s_{iq} = s_{jq} + \beta_1 c$. Откуда с учётом (3) и того факта, что $s_{ij} = -s_{ji} + \beta_2 c$, получаем

$$a_q = \sigma^{s_{jq} + \beta c}(a_j).$$

Поскольку $a_j \in M_{c,\sigma}$, то $\sigma^c(a_j) = a_j$, следовательно, для любого α справедливо

$$a_q = \sigma^{s_{jq}+\alpha c}(a_j).$$

Как мы уже говорили, по следствию из леммы 10 получаем, что между вершинами v_{y_j} и v_{y_q} в графе G_F существуют только пути с длинами $s_{jq} + \alpha c$. Принимая это во внимание, а также последнее равенство и тот факт, что это справедливо для любой пары вершин графа G_F, получаем, что при присвоенных выше переменным предиката p значениях каждое слагаемое конъюнкции в формуле F будет истинным, а значит $p(\tilde{a}) = TRUE$.

Лемма доказана.

Докажем теперь основную теорему данной работы.

Теорема 2. В решётке замкнутых классов k-значной логики над классом самодвойственных функций S_σ находятся только классы семейств $\mathbf{S}_\sigma$, $\mathbf{T}_\sigma$, их пересечения и P_k.

Доказательство Из лемм 3 и 7 следует, что указанные семейства классов содержат класс S_σ. Поэтому нам достаточно показать, что любой замкнутый класс A, содержащий класс S_σ, совпадает с одним из классов, указанных в формулировке леммы.

Рассмотрим один из предикатов p', который сохраняют все функции системы A, то есть $p' \in \mathrm{Inv}(A)$. Из того, что $S_\sigma \subseteq A$ следует, что все функции S_σ сохраняют предикат p', то есть

$$p' \in \mathrm{Inv}(S_\sigma). \tag{4}$$

Из $(*)$ следует, что $\mathrm{Pol}([R_\sigma]) = \mathrm{Pol}(R_\sigma) = S_\sigma$. По определению класс $[R_\sigma]$ — замкнутый. Покажем, что он содержит все диагонали. В самом деле, диагональ $d(x_1, x_2) = (x_1 = x_2)$ получается следующим образом: $d(x_1, x_2) = R_{\sigma^h}(x_1, x_2)$ (используем лемму 3). Остальные диагонали принадлежат классу $[d(x_1, x_2)]$. Следовательно, согласно (1)

$$[R_\sigma] = \mathrm{Inv}(\mathrm{Pol}([R_\sigma])) = \mathrm{Inv}(S_\sigma).$$

Используя (4), получаем $p' \in [R_\sigma]$.

Сопоставим некоторой формуле F', задающей предикат p' над $[R_\sigma]$, с вынесенными вперёд кванторами существования граф $G_{F'}$ (как описано выше). Пусть граф $G_{F'}$ имеет t компонент связности $G_1, \ldots, G_t$. Каждая компонента связности G_i очевидным образом задаёт свой предикат p_i. По определению графа предиката получаем, что $p' = p_1 \& p_2 \& \ldots \& p_t$ (поскольку разные предикаты p_i и p_j не имеют общих переменных), $p_i \in [R_\sigma]$.

Лемма 12([1]). Если $R = R_1 \& R_2$, то $A_R = A_{R_1} \cap A_{R_2}$.

Нам остаётся показать, что предикат p_i $(i = 1, \ldots, t)$ задаёт весь P_k (то есть $\mathrm{Pol}(p_i) = P_k$) либо класс из семейства $\mathbf{S}_\sigma$ или $\mathbf{T}_\sigma$. Действительно, из последней леммы будет следовать, что предикат p' задаёт класс из указанных семейств или пересечение подобных классов. Поскольку данное утверждение справедливо для произвольного предиката из $\mathrm{Inv}(A)$, то по определению функторов Inv и Pol класс A обязан совпадать с одним из классов, указанных в формулировке леммы.

Для удобства изложения обозначим через p произвольный сомножитель p_i из конъюнкции, записанной выше, задающей предикат p'; G_F — соответствующий p подграф графа $G_{F'}$.

Перенумеруем вершины графа G_F. Опять будем предполагать, что первые n вершин $v_{y_1}, \ldots, v_{y_n}$ $(n \geqslant 1)$ графа G_F соответствуют свободным переменным, остальные — связанным. Составим для G_F матрицу расстояний D_F (по определению 9). Если граф содержит циклы с ненулевыми длинами $c_1, \ldots, c_q$, обозначим $c = \mathrm{HOD}(c_1, \ldots, c_q)$. Если длины всех циклов кратны h, положим $c = h$. Оставим в каждом множестве, являющемся элементом матрицы расстояний D_F, по одному минимальному неотрицательному элементу s_{ij}. Согласно следствию из леммы 10, мы вычеркнем из множеств элементы вида $s_{ij} + \alpha c$. Для простоты рассуждений далее будем считать, что элементы матрицы D_F являются числами $d_{ij} = s_{ij}$. Из определения матрицы D_F следует, что $d_{ij} = c - d_{ji}$, $d_{ii} = 0$.

Введём множество $M_{c,\sigma}$, как описано выше. Отметим, что в случае $c = h$ будет справедливо $M_{c,\sigma} = E_k$. Обозначим через π подстановку, определённую на множестве $M_{c,\sigma}$ и совпадающую на нём с подстановкой σ. В случае $c = h$ будет выполнено $\pi = \sigma$.

Рассмотрим теперь следующие случаи.

Пусть предикат p — одноместный ($n = 1$). По лемме 11 получаем, что $p(x) = TRUE$ тогда и только тогда, когда $x \in M_{c,\sigma}$. Получаем, что $\mathrm{Pol}(p) = P_k$, либо замкнутый класс $\mathrm{Pol}(p)$ является классом сохранения множества, а предикат p — одноместным центральным предикатом. В последнем случае $\mathrm{Pol}(p)$ входит в семейство $\mathbf{T}_\sigma$.

Пусть предикат p — двухместный ($n = 2$). Из леммы 11 следует, что $\mathrm{Pol}(p) = H_{c,\pi^{d_{12}}}$. В случае, когда $c = h$, получаем следующее

$$\mathrm{Pol}(p) = \begin{cases} S_{\sigma^{d_{12}}}, & d_{12} \neq 0, \\ P_k, & d_{12} = 0. \end{cases}$$

Пусть $n > 2$. Покажем, что в данном случае существует двухместный предикат R' такой, что $p \in [R']$, $R' \in [p]$, то есть $\mathrm{Pol}(p) = \mathrm{Pol}(R')$ (по $(*)$), а предикат R' относится ко второму случаю. Этим теорема будет доказана.

Рассмотрим первую строку матрицы D_F. Если найдётся два равных элемента d_{1i} и d_{1j}, то, согласно лемме 11, в любом наборе $\tilde{a} \in E_k^n$, удовлетворяющем предикату p, компоненты a_i и a_j равны. Используя диагональ, можно показать, что мы можем устранить одну из этих переменных, не изменив класс, задаваемый предикатом. Поэтому далее будем считать, что все элементы d_{1i} различны.

Применив алгоритм Евклида ([7]), получим, что существуют целые числа $b_1, \ldots, b_n$ такие, что

$$d = \mathrm{HOD}(d_{11}, \ldots, d_{1n}) = \sum_{i=1}^{n} b_i d_{1i}. \tag{5}$$

Обозначим $R'(x_1, x_2) = R_{\pi^d}(x_1, x_2)$.

Пусть $d_{1i} = s_i d$. Тогда:

$$R_{\pi^{d_{1i}}}(x_1, x_2) = R_{\pi^{s_i d}}(x_1, x_2) = \exists y_1, \ldots, y_{s_i - 1} R_{\pi^d}(x_1, y_1) \& R_{\pi^d}(y_1, y_2) \& \ldots \& R_{\pi^d}(y_{s_i - 1}, x_2).$$

Граф формулы в правой части равенства G_1 получается соединением графов формул, задающих предикат R_{π^d}. Поскольку указанное соединение происходит по одной вершине, то в графе G_1 будут только циклы с длинами αc. В силу леммы 11, указанная формула действительно будет задавать предикат $R_{\pi^{d_{1i}}}$. Введём следующий предикат:

$$\hat{p}(x_1, \ldots, x_n) = R_{\pi^{d_{12}}}(x_1, x_2) \& R_{\pi^{d_{13}}}(x_1, x_3) \& \ldots \& R_{\pi^{d_{1n}}}(x_1, x_n).$$

Аналогично предыдущему случаю граф формулы (обозначим её $\hat{F}$), задающей предикат $\hat{p}$, обладает только циклами длины αc. Заметим, что в графе $G_{\hat{F}}$ существует путь между вершинами u_{y_j} и u_{y_q}, соответствующими свободным переменным (то есть $j, q \leqslant n$), длины равной $d_{j1} + d_{1q}$. Поскольку в графе G_F есть путь между вершинами v_{y_j} и v_{y_q} указанной длины, то в силу следствия из леммы 10, получаем, что существует число β такое, что $d_{j1} + d_{1q} = \beta c + d_{jq}$. Поскольку в качестве числа d_{jq} мы выбрали минимальное неотрицательное расстояние между рассматриваемыми вершинами в графе G_F, то в силу последнего равенства, следствия из леммы 10 и того факта, что в графе $G_{\hat{F}}$ есть только циклы длины αc, получаем, что d_{jq} является также минимальным неотрицательным расстоянием между вершинами u_{y_j} и u_{y_q} в графе $G_{\hat{F}}$. Следовательно, матрицы расстояний графов G_F и $G_{\hat{F}}$ совпадают. По лемме 11 получаем, что $p = \hat{p}$. Иными словами, мы показали, что $p \in [R']$.

Справедливость следующего соотношения вытекает непосредственно из леммы 11.

$$R_{\pi^{b_i d_{1i}}}(x_1, x_2) = \exists y_j^i p(x_1, y_2^1 \ldots, y_{i-1}^1, y_i^1, y_{i+1}^1, \ldots, y_n^1) \& p(y_i^1, y_2^2 \ldots, y_{i-1}^2, y_i^2, y_{i+1}^2, \ldots, y_n^2) \&$$

$$\& p(y_i^2, y_2^3 \ldots, y_{i-1}^3, y_i^3, y_{i+1}^3, \ldots, y_n^3) \& \& \ldots \& p(y_i^{b_i - 1}, y_2^{b_i} \ldots, y_{i-1}^{b_i}, x_2, y_{i+1}^{b_i}, \ldots, y_n^{b_i}),$$

где под символом $\exists y_j^i$ подразумевается существование всех переменных, имеющих вид y_j^i и встречающихся в формуле.

Далее в силу (5) получаем

$$R_{\pi^d}(x_1, x_2) = \exists y_1, \ldots, y_{n-1} R_{\pi^{b_1 d_{11}}}(x_1, y_1) \& R_{\pi^{b_2 d_{12}}}(y_1, y_2) \& \ldots \& R_{\pi^{b_n d_{1n}}}(y_{n-1}, x_2).$$

Итак, $R' \in [p]$.

Теорема доказана.

5 Основные свойства решётки замкнутых классов над S_σ.

Обозначим для краткости через $S_{i,j}$ замкнутый класс H_{j,π^i} (см. определение 7) в случае, если $M_{j,\sigma} \neq E_k$, и класс S_{σ^i} в ином случае. При этом будем считать, что число j равно наименьшему общему кратному длин циклов исходной подстановки σ, элементы которых входят в множество $M_{j,\sigma}$, а число i — делитель минимального t такого, что $\sigma^t = e$ на множестве $M_{j,\sigma}$.

Лемма 13. Пусть $M_{l_1,\sigma} \subseteq E_k$. Для любого i_1 — делителя минимального t такого, что $\sigma^t = e$ на множестве $M_{l_1,\sigma}$, причём $\sigma^{i_1} \neq e$ на множестве $M_{l_1,\sigma}$, i_2 такого, что $\mathrm{HOD}(l_1, i_2)$ не делит i_1, l_2 такого, что справедливы включения $M_{l_1,\sigma} \subseteq M_{l_2,\sigma} \subseteq E_k$ $(l_1|l_2)$ существует функция $f^\sigma_{i_1,i_2,l_1,l_2}$ такая, что:

1. $f^\sigma_{i_1,i_2,l_1,l_2} \notin S_{i_1,l_1}$;
2. Для любого l_3: $M_{l_1,\sigma} \not\subseteq M_{l_3,\sigma}$, m — делителя минимального t такого, что $\sigma^t = e$ на множестве $M_{l_3,\sigma}$ выполнено $f^\sigma_{i_1,i_2,l_1,l_2} \in S_{m,l_3}$;
3. Для любого r: $i_2 r$ делит минимальное t такое, что $\sigma^t = e$ на множестве $M_{l_2,\sigma}$, выполнено $f^\sigma_{i_1,i_2,l_1,l_2} \in S_{i_2 r,l_2}$.

Доказательство Обозначим $\pi_1 = \sigma^{i_1}$, $\pi_2 = \sigma^{i_2}$. Рассмотрим множество $M_{l_1,\sigma}$. Пусть оно состоит из элементов, входящих в циклы подстановки π_1 с длинами $m_1, \ldots, m_p$. Возьмём в каждом из указанных циклов по произвольному элементу $a_1, \ldots, a_p$, пусть $b_q = \pi_1(a_q)$, $q = 1, \ldots, p$.

Поскольку подстановка $\pi_1 \neq e$ на множестве $M_{l_1,\sigma}$, у π_1 на указанном множестве существует цикл длины, большей 1. Пусть элемент d принадлежит этому циклу.

Обозначим через $f^\sigma_{i_1,i_2,l_1,l_2}$ p-местную функцию, равную d на наборах $\tilde{a} = (a_1, \ldots, a_p)$ и $\tilde{b} = (b_1, \ldots, b_p)$, на любом наборе $\tilde{c} = (c_1, \ldots, c_p)$ таком, что существует число m: $c_q = \pi_2^m(a_q)$ для всех $q = 1, \ldots, p$ или $c_q = \pi_2^m(b_q)$ для указанных q, значение функции равно $\pi_2^m(d)$ (в дальнейшем множество наборов данного вида вместе с наборами $\tilde{a}$ и $\tilde{b}$ будем обозначать C), на остальных наборах $f^\sigma_{i_1,i_2,l_1,l_2}(\tilde{x}) = x_1$.

Покажем, что указанная функция определена корректно.

Во-первых, необходимо доказать, что не существует чисел $\alpha \neq \beta$ таких, что $\pi_2^\alpha(\tilde{a}) = \pi_2^\beta(\tilde{b})$. Предположим противное, тогда

$$\sigma^{i_2(\alpha-\beta)}(a_q) = b_q \Leftrightarrow \sigma^{i_2(\alpha-\beta)-i_1}(a_q) = a_q,$$

для всех $q = 1, \ldots, p$. Предположим, что элементы $a_1, \ldots, a_p$ входят в циклы исходной подстановки σ с длинами $m'_1, \ldots, m'_{p'}$, откуда $m'_j | (i_2(\alpha - \beta) - i_1)$. А поскольку мы условились $l_1 = \mathrm{HOK}(m'_1, \ldots, m'_{p'})$, то $l_1 | (i_2(\alpha - \beta) - i_1)$. Следовательно, существуют константы k_1, k_2 такие, что справедливо:

$$k_1 i_2 + k_2 l_1 = i_1,$$

откуда $\mathrm{HOD}(l_1, i_2)$ делит i_1. Получаем противоречие с условием леммы.

Во-вторых, для любого t такого, что $\pi_2^t(a_q) = a_q$ для всех $q = 1, \ldots, p$ (все такие t кратны числу $\mathrm{HOK}(m_1, \ldots, m_p)$) должно выполняться

$$d = f(\tilde{a}) = f(\pi_2^t(a_1), \ldots, \pi_2^t(a_p)) = \pi_2^t(d).$$

Несложно проверить, что это условие обеспечивается выбором элемента d.

Покажем теперь, что построенная функция удовлетворяет сформулированным условиям. Напомним, что рассматриваемые классы $S_{i,j}$ задаются двухместными предикатами, описанными в определении 7. Для краткости обозначим указанные предикаты через $p_{i,j}$ и будем считать, что они задаются формулами, как описано в лемме 7.

Рассмотрение наборов $\tilde{a} = (a_1, \ldots, a_p)$ и $\tilde{b} = (b_1, \ldots, b_p)$ для предиката p_{i_1,l_1} даёт $f^\sigma_{i_1,i_2,l_1,l_2} \notin S_{i_1,l_1}$.

Пусть $M_{l_1,\sigma} \not\subseteq M_{l_3,\sigma}$. Следовательно, существует цикл C_t подстановки σ^{i_1} длины m_t элементы которого принадлежат множеству $M_{l_1,\sigma}$ и не принадлежат $M_{l_3,\sigma}$. Получаем, что не принадлежат множеству $M_{l_3,\sigma}$ и элементы цикла C'_t исходной подстановки σ, измельчением которого получен цикл C_t. Рассмотрим любые два p-местных набора $\tilde{x}$ и $\tilde{y}$, покомпонентно удовлетворяющие предикату p_{m,l_3}, где m — любое число. Из сказанного выше и того факта, что t-я компонента любого набора из множества C принадлежит циклу C_t следует, что $\tilde{x}, \tilde{y} \notin C$. Получаем, что $(f^{\sigma}_{i_1,i_2,l_1,l_2}(\tilde{x}), f^{\sigma}_{i_1,i_2,l_1,l_2}(\tilde{y})) = (x_1, y_1) \in p_{m,l_3}$. Следовательно, $f^{\sigma}_{i_1,i_2,l_1,l_2} \in S_{m,l_3}$.

Пусть наконец $M_{l_1,\sigma} \subseteq M_{l_2,\sigma} \subseteq E_k$, i_2, r удовлетворяют условиям леммы. Снова возьмём два произвольных набора $\tilde{x}$, $\tilde{y}$, покомпонентно удовлетворяющих предикату p_{i_2r,l_2}.

Предположим, что $\tilde{x} \in C$. Следовательно, существует m: $\pi^m_2(\tilde{a}) = (\pi^m_2(a_1), \ldots, \pi^m_2(a_p)) = \tilde{x}$, откуда $\tilde{y} = \pi^{m+r}_2(\tilde{a})$, либо указанные соотношения справедливы для набора $\tilde{b}$. В любом случае получаем, что $\tilde{y} \in C$. Итак, либо оба набора $\tilde{x}$, $\tilde{y}$ принадлежат множеству C, либо оба не принадлежат C. Во втором случае утверждение $f^{\sigma}_{i_1,i_2,l_1,l_2} \in S_{i_2r,l''}$ очевидно. В первом случае:

$$p_{i_2r,l_2}(f^{\sigma}_{i_1,i_2,l_1,l_2}(\tilde{x}), f^{\sigma}_{i_1,i_2,l_1,l_2}(\tilde{y})) = p_{i_2r,l_2}(f^{\sigma}_{i_1,i_2,l_1,l_2}(\pi^m_2(\tilde{a})), f^{\sigma}_{i_1,i_2,l_1,l_2}(\pi^{m+r}_2(\tilde{a})) =$$

$$= p_{i_2r,l_2}(d', \pi^r_2(d')) = TRUE,$$

где $d' = \pi^m_2(d)$.

Лемма доказана.

Лемма 14. Пусть $M_{l_1,\sigma} \subset E_k$. Для любого i_1 — делителя минимального t такого, что $\sigma^t = e$ на множестве $M_{l_1,\sigma}$, l_2 такого, что справедливы включения $M_{l_1,\sigma} \subset M_{l_2,\sigma} \subseteq E_k$ $(l_1|l_2)$, и существует цикл $C' \subseteq M_{l_2,\sigma} \setminus M_{l_1,\sigma}$ подстановки σ длины l', удовлетворяющей для некоторого i_2 — делителя минимального t такого, что $\sigma^t = e$ на множестве $M_{l_2,\sigma}$ условию $l'|\,\mathrm{НОК}(i_2, l_1)$, существует функция $g^{\sigma}_{i_1,C',l_1}$ такая, что:

1. $g^{\sigma}_{i_1,C',l_1} \notin S_{i_1,l_1}$;
2. Для любого l_3: $M_{l_1,\sigma} \not\subseteq M_{l_3,\sigma}$, m — делителя минимального t такого, что $\sigma^t = e$ на множестве $M_{l_3,\sigma}$ выполнено $g^{\sigma}_{i_1,C',l_1} \in S_{m,l_3}$;
3. $g^{\sigma}_{i_1,C',l_1} \in S_{i_2,l_2}$.

Доказательство Доказательство очень похоже на предыдущее. Аналогичным образом выбираем набор $\tilde{a}$, обозначим $\pi_1 = \sigma^{i_1}$, $\pi_2 = \sigma^{i_2}$. Пусть элемент $d \in C'$. Определим p-местную функцию $g^{\sigma}_{i_1,C',l_1}$ следующим образом. Положим $g^{\sigma}_{i_1,C',l_1}(\tilde{a}) = d$; на любом наборе $\tilde{c} = (c_1, \ldots, c_p)$ таком, что существует число m: $c_q = \pi^m_2(a_q)$ для всех $q = 1, \ldots, p$, функция $g^{\sigma}_{i_1,C',l_1}$ равна $\pi^m_2(d)$. Опять обозначим указанное множество p-местных наборов (вместе с $\tilde{c}$) через C. На всех остальных наборах положим $g^{\sigma}_{i_1,C',l_1}(\tilde{x}) = x_1$.

Проверим корректность определения функции. В отличие от предыдущего доказательства, здесь достаточно проверить только второе условие.

Пусть число r — минимальное, удовлетворяющее условию $\pi^r_2(a_q) = a_q$ для всех $q = 1, \ldots, p$. Рассуждая аналогично доказательству леммы 13, получаем, что $ri_2 = tl_1 = \mathrm{НОК}(i_2, l_1)$.

Получаем:

$$d = f(\tilde{a}) = f(\pi^r_2(a_1), \ldots, \pi^r_2(a_p)) = \pi^r_2(d) = \sigma^{ri_2}(d).$$

Указанное выражение выполнено тогда и только тогда, когда элемент d принадлежит циклу подстановки σ, длина которого l' делит число ri_2. Поскольку проведённые рассуждения справедливы и в обратную сторону, получаем, что корректность определения функции g_{i_1,C',l_1} обеспечивается условием леммы.

Покажем, что функция $g^{\sigma}_{i_1,C',l_1}$ удовлетворяет сформулированным требованиям.

Аналогично предыдущему случаю, рассмотрение наборов $\tilde{a}$ и $\tilde{b} = (\pi_1(a_1), \ldots, \pi_1(a_p))$ (включая случай $\tilde{a} = \tilde{b}$, когда S_{i_1,l_1} — центральный класс) даёт $g^{\sigma}_{i_1,C',l_1} \notin S_{i_1,l_1}$.

Доказательство второго и третьего пунктов полностью повторяет аналогичные доказательства из предыдущей леммы.

Лемма доказана.

Следующая теорема целиком описывает решётку замкнутых классов над произвольным самодвойственным классом.

Теорема 3. $S_{i_1,l_1} \subseteq S_{i_2,l_2}$ тогда и только тогда, когда выполнены следующие условия:

1. $M_{l_2,\sigma} \subseteq M_{l_1,\sigma}$ (при нашем соглашении это эквивалентно $l_2|l_1$).
2. $\mathrm{HOD}(i_1,l_2)|i_2$.
3. Для любого цикла C' длины l' подстановки σ на множестве $M_{l_1,\sigma} \setminus M_{l_2,\sigma}$ число l' не делит $\mathrm{HOK}(i_1,l_2)$.

Доказательство Если нарушается первое требование, то существует цикл C_1 подстановки σ, принадлежащий множеству $M_{l_2,\sigma} \setminus M_{l_1,\sigma}$. В этом случае достаточно рассмотреть функцию одного переменного $u(x)$, являющуюся тождественной на $E_k \setminus C_1$ и равной некоторой константе $d \in M_{l_1,\sigma}$ на множестве элементов цикла C_1. Очевидно, $u(x) \in S_{i_1,l_1} \setminus S_{i_2,l_2}$.

Если нарушается третье требование, то выполняются все условия леммы 14. Следовательно, существует функция $g^{\sigma}_{i_2,C',l_2} \in S_{i_1,l_1} \setminus S_{i_2,l_2}$. Если нарушается второе требование, и S_{i_2,l_2} не является центральным классом (то есть, подстановка σ^{i_2} не является тождественной на множестве $M_{l_2,\sigma}$), то выполнены условия леммы 13, и существует функция $f^{\sigma}_{i_2,i_1,l_2,l_1} \in S_{i_1,l_1} \setminus S_{i_2,l_2}$.

Пусть теперь класс S_{i_2,l_2} — центральный и нарушено только второе условие. Аналогично предыдущим доказательствам возьмём набор $\tilde{a}$, где a_i принадлежит i-му циклу подстановки σ^{i_2} длины m_i $(i = 1,\ldots,p)$. Если $M_{l_1,\sigma} = M_{l_2,\sigma}$, то второе условие выполнено, поскольку любой класс вида S_{t,l_1} вложен в центральный класс с центром $M_{l_1,\sigma}$. Следовательно, существует $d \in M_{l_1,\sigma} \setminus M_{l_2,\sigma}$.

Определим далее функцию h. Положим $h(\tilde{a}) = d$, для любого набора $\tilde{c}$ такого, что для некоторого m справедливо $c_i = \sigma^{i_1 m}(a_i)$ положим $h(\tilde{c}) = \sigma^{i_1 m}(d)$. На остальных наборах, как и раньше, положим $h(\tilde{x}) = x_1$. Корректность этого определения показывается аналогично доказательству леммы 14 (в силу справедливости третьего условия). Несложно проверить, что $h \in S_{i_1,l_1} \setminus S_{i_2,l_2}$.

С другой стороны, пусть перечисленные в формулировке условия выполнены.

Напомним, что мы обозначили через $p_{i,j}$ двухместные предикаты из определения 7, задающие замкнутые классы $S_{i,j}$. Далее мы покажем, что предикат p_{i_2,l_2} можно получить формулой F над $[p_{i_1,l_1}]$. Сделаем это аналогично доказательству леммы 7: граф формулы F будет состоять из петли длины d_1 и ориентированного пути длины d_2.

Возьмём $d_1 = \frac{\mathrm{HOK}(i_1,l_2)}{i_1}$. Если мы рассмотрим формулу, реализующую p_{i_2,l_2} над $[R_\sigma]$, то длина построенной петли в графе указанной формулы (обозначим её F') равна $d_1 i_1 = \mathrm{HOK}(i_1,l_2)$. Третье условие формулировки теоремы обеспечивает нам то, что элементы из множества $M_{l_1,\sigma} \setminus M_{l_2,\sigma}$ не войдут в область определения нового предиката.

Из второго условия и алгоритма Евклида следует, что существуют целые числа t_1, t_2, t_3 такие, что выполнено

$$i_2 = t_1 \,\mathrm{HOD}(i_1,l_2) = t_2 i_1 + t_3 l_2.$$

Возьмём в качестве d_2 число t_2. Тогда в формуле F' длина ориентированного пути будет равна $d_2 i_1 = i_2 \mod (l_2)$.

Итак, с учётом вышесказанного и леммы 11:

$$p_{i_2,l_2}(x_1,x_2) = \exists y_1,\ldots,y_{d_1+d_2-2} p_{i_1,l_1}(x_1,y_1)\&p_{i_1,l_1}(y_1,y_2)\&\ldots\&p_{i_1,l_1}(y_{d_1-1},x_1)\&$$

$$\&p_{i_1,l_1}(x_1,y_{d_1})\&p_{i_1,l_1}(y_{d_1},y_{d_1+1})\&\ldots\&p_{i_1,l_1}(y_{d_1+d_2-2},x_2).$$

Теорема доказана

Рассмотрим теперь вопрос о виде пересечений классов, указанных в теореме 2.

Лемма 15. Пересечения классов вида $S_{i_1,j_1} \cap S_{i_2,j_2} \cap \ldots \cap S_{i_{s_1},j_{s_1}}$ и $S_{i'_1,j_1} \cap S_{i'_2,j_2} \cap \ldots \cap S_{i'_{s_2},j_{s_2}}$ различны если:

1. Все множества $M_{j_i,\sigma}$ различны.

2. Для любого $M_{j_l,\sigma}$: если каждое из множеств $M_{j_{l_1},\sigma},\dots M_{j_{l_t},\sigma}$ содержит $M_{j_l,\sigma}$, то $\mathrm{HOD}(i_0,j_l)$ не делит i_l, где $i_0=\mathrm{HOD}(i_{l_1},\dots,i_{l_t})$, или существует цикл C' подстановки σ^m на множестве $M_0=\bigcap_{r=1}^t M_{j_{l_r},\sigma}$ длины l', делящей $\mathrm{HOK}(i_l,c_{i_l,j_l,\sigma})$.

Для любого пересечения указанного вида, не обладающего перечисленными выше двумя свойствами существует равное ему пересечение, обладающее этими свойствами.

Доказательство Докажем вначале вторую часть леммы.

Покажем, что пересечение классов из семейства $\mathbf{S}_\sigma$ даёт класс из этого же семейства. Возьмём два различных делителя i,j числа h. Пусть $q=\mathrm{HOD}(i,j)$. Покажем, что $S_{\sigma^i}\cap S_{\sigma^j}=S_{\sigma^q}$.

По лемме 12 получаем, что рассматриваемое пересечение задаёт предикат $R(x,y,z,u)=R_{\sigma^i}(x,y)R_{\sigma^j}(z,u)$. Применим рассуждения из доказательства теоремы 2 для предиката $R'(x,y,u)=R(x,y,x,u)$. В матрице расстояний D_F формулы, задающей этот предикат, в первой строке будут числа $d_{12}=i$, $d_{13}=j$. Согласно рассмотренному в доказательстве теоремы 2 третьему случаю справедливо $\mathrm{Pol}(R_{\sigma^q})=\mathrm{Pol}(R')=\mathrm{Pol}(R)=S_{\sigma^q}$.

Аналогичное рассуждение справедливо и для пересечения классов вида H_{d,π^i}:

$$H_{d,\pi^i}\cap H_{d,\pi^j}=H_{d,\pi^q},$$

где $q=\mathrm{HOD}(i,j)$. Следовательно, в формуле, задающей пересечение, мы можем заменить классы с равными множествами $M_{j_i,\sigma}$ на один класс.

Предположим теперь, что нарушено второе условие леммы.

По лемме 12 получаем, что класс $S_{i_{l_1},j_{l_1}}\bigcap\dots\bigcap S_{i_{l_p},j_{l_p}}$ можно задать предикатом

$$R'(x_1,\dots,x_{2p})=p_{i_{l_1},j_{l_1}}(x_1,x_2)\&\dots\&p_{i_{l_p},j_{l_p}}(x_{2p-1},x_{2p}).$$

Рассмотрим предикаты вида:

$$R'_r(x_1,x_{2r})=\exists x_2x_4\dots x_{2(r-1)}x_{2(r+1)}\dots x_{2p}p_{i_{l_1},j_{l_1}}(x_1,x_2)\&\dots\&p_{i_{l_p},j_{l_p}}(x_1,x_{2p})$$

(мы отождествили в формуле все переменные с нечётным индексом, обозначив получившуюся при этом переменную x_1, и провели операции проекций по всем переменным кроме x_1 и какой-либо ещё переменной). Получившиеся предикаты задают классы $S_{i_{l_r},m_0}$, где $m_0=\mathrm{HOD}(j_{l_1},\dots,j_{l_t})$ или $M_{m_0,\sigma}=M_0$. По теореме 1 получаем

$$\bigcap_{r=1}^t S_{i_{l_r},j_{l_r}}\subseteq S_{i_0,m_0}.$$

Откуда получаем (в силу последней теоремы), что при нарушении второго условия леммы

$$\bigcap_{r=1}^t S_{i_{l_r},j_{l_r}}\subseteq S_{i_l,j_l}.$$

Это означает, что мы можем вычеркнуть слагаемое S_{i_l,j_l}.

Докажем теперь первую часть.

Рассмотрим два пересечения, описанные в формулировке леммы. Поскольку мы предполагаем, что формулы, задающие эти пересечения различны, найдётся номер q такой, что $i_q\neq i'_q$ или один из классов S_{i_q,j_q}, $S_{i'_q,j'_q}$ отсутствует в соответствующей формуле. Пусть множества $M_{j_{l_1},\sigma},\dots M_{j_{l_t},\sigma}$ содержат множество $M_{j_q,\sigma}$. Если найдётся номер $p=1,\dots,t$ такой, что $i_{l_p}\neq i'_{l_p}$ или один из классов S_{i_q,j_q}, $S_{i'_q,j'_q}$ отсутствует в соответствующей формуле, то в качестве номера q возьмём l_p (снова проверив указанное условие).

Согласно выбору q $i_{l_1}=i'_{l_1},\dots,i_{l_t}=i'_{l_t}$. Рассмотим два класса S_{i_q,j_q} и $S_{i'_q,j'_q}$. Если один из них отсутствует в своей формуле, задающей пересечение, то обозначим оставшийся через $S_1=S_{t_1,r_1}$, в ином случае (то есть, если оба указанных класса присутствуют в формулах),

если $S_{i_q,j_q} \not\subseteq S_{i'_q,j'_q}$, положим $S_1 = S_{i_q,j_q}$, $t_1 = i_q$, $r_1 = j_q$, $S_2 = S_{i'_q,j'_q}$, $t_2 = i'_q$, $r_2 = j'_q$, если же $S_{i_q,j_q} \subseteq S_{i'_q,j'_q}$, то обозначим наоборот: $S_1 = S_{i'_q,j'_q}$, $t_1 = i'_q$, $r_1 = j'_q$, $S_2 = S_{i_q,j_q}$, $t_1 = i'_q$, $r_1 = j'_q$.

Из условия данной леммы следует, что применением леммы 13 или 14 мы получим функцию $f \in S_{\mathrm{HOD}(i_{l_1},\ldots,i_{l_t},t_2),\mathrm{HOD}(j_{l_1},\ldots,j_{l_t},r_2)} \setminus S_{t_1,r_1}$, $f \in S_{i_q,j_q}$, если $q \notin \{l_1,\ldots,l_t\}$.

Поскольку выполняется второе условие леммы, возможны два варианта.

Если $\mathrm{HOD}(i',j_l)$ не делит i_l, где $i' = \mathrm{HOD}(i_{l_1},\ldots,i_{l_t},t_2)$, то $\mathrm{HOD}(i_{l_q},j_l)$ не делит i_l $(q = 1,\ldots,t)$ и $\mathrm{HOD}(t_2,j_l)$ не делит i_l. По лемме 13 получаем, что функция f принадлежит классам $S_{i_{l_q},j_{l_q}}$ $(q = 1,\ldots,t)$, S_2.

Если же существует указанный в формулировке цикл C' длины l', делящей $\mathrm{HOK}(i',j_l)$, то l' тем более делит все числа $\mathrm{HOK}(i_{l_q},j_l)$ $(q = 1,\ldots,t)$ и $\mathrm{HOK}(t_2,j_l)$. По лемме 14 получаем, что функция f принадлежит классам $S_{i_{l_q},j_{l_q}}$ $(q = 1,\ldots,t)$, S_2.

В любом случае функция f не принадлежит одному пересечению (в которое входит класс S_1) и не принадлежит другому.

Лемма доказана.

Из указанной леммы следует, что число рассматриваемых пересечений в общем случае достаточно велико.

Работа выполнена при поддержке РФФИ, грант 09-01-00701.

Список литературы

[1] Яблонский С. В., Гаврилов Г. П., Набебин А. А., Предполные классы в многозначных логиках. Издательский дом МЭИ, Москва, 1997.

[2] Rosenberg I., La structure des fonctions de plusieur variables sur un ensemble fini. Comptes Rendus, de l'Academ. Paris, 260, 1965, 3817-3819.

[3] Яблонский С. В., Функциональные построения в k-значной логике. Труды математического института им. Стеклова АН СССР, 1958, т. 51, 100-109.

[4] Бондарчук В. Г., Калужнин В. А., Котов В. Н., Ромов Б. А., Теория Галуа для алгебр Поста. Кибернетика, 1969, №3, 1-10, №5, 1-9.

[5] Марченков С. С., Замкнутые классы булевых функций. Физматлит, Москва, 2000, 126 с.

[6] Марченков С. С., Клоновая классификация дуально дискриминаторных алгебр с конечным носителем. Математические заметки, т. 61, вып. 3, 1997, 359-366.

[7] Арнольд И. В., Теоретическая арифметика. Учпедгиз, Москва, 1938, 480 с.

УДК 004.65

РЕЛЯЦИОННЫЕ ДАННЫЕ В ИНТЕГРИРОВАННЫХ СРЕДСТВАХ СЕМАНТИЧЕСКОГО WEB

© 2009 г. Д. В. Левшин

levshin@nicevt.ru

Кафедра Системного Программирования

1 Введение

Концепция Семантического Web была представлена научной общественности автором Всемирной паутины (WWW) Тимом Бенерс-Ли в 2001 году [1]. Перспективы использования машинного понимания информации в Интернет, интеллектуальных агентов для решения повседневных задач потребителей, обрисованные в данной статье, были вдохновляющими. Поэтому с этого момента интерес в академических кругах к Семантическому Web постоянно возрастал, что выражалось в экспоненциальном росте числа публикаций, разработок и конференций в данной области. Однако долгое время все появляющиеся результаты так и оставались лишь научными "игрушками", что порождало скептическое отношение к Семантическому Web [2]. Поэтому важным является то, что в последнее время стали появляться реальные системы, использующие семантические технологии; добавление этих технологий во Всемирную паутину получило название Web 3.0 [3, 4].

Условия Интернет предъявляют строгие требования к производительности и масштабируемости средств поддержки семантических форматов, которые могут использоваться при разработке web-приложений третьего поколения. В частности, RDF-хранилища должны предоставлять доступ к большим объемам RDF-троек; системы рассуждений должны уметь выполнять логический вывод утверждений на основе баз знаний, содержащих большое число RDF-утверждений и OWL-описаний. Реляционные базы данных за последние десятилетия стали ведущей технологией хранения больших объемов информации и выполнения к ней доступа, поэтому для разработки RDF-хранилищ и систем рассуждений используются реляционные системы управления баз данных (РСУБД).

Другим важным аспектом взаимосвязи технологий баз данных и Семантического Web является наполнение Семантического Web реляционными данными. Большая доля информации во Всемирной паутине (так называемый Глубокий Web) хранится в базах данных [5], и невозможность обращения к этой информации из семантических систем стала бы барьером к их использованию.

В данной статье рассматривается, каким образом RDF-хранилища и системы рассуждений, основанные на использовании РСУБД, могут обеспечить доступ к реляционным данным на примере [6].

2 Использование СУБД для поддержки форматов Семантического Web

Консорциум W3C разработал стек языков Семантического Web, направленных на машинную обработку и понимание информации. С подробным описанием языков можно ознакомиться в [7], автор также рассматривал их в [8]. Поэтому лишь отметим основные форматы. Язык RDF является основой Семантического Web и определяет его модель данных. Языки RDF Schema и OWL расширяют RDF возможностью записи логических утверждений относительно концепций (классов ресурсов) и индивидуумов. Таким образом, средства поддержки форматов Семантического Web должны позволять работу и с явно заданными утверждениями, и выведенными в результате логических рассуждений. Язык запросов SPARQL разработан для доступа к RDF-данным.

Первые средства для работы с RDF-данными и выполнения рассуждений на основе OWL выполняли обработку информации в основной памяти. В частности, можно назвать открытую систему рассуждений Pellet [9]. С ростом объемов RDF-данных и размеров OWL-онтологий стала очевидна необходимость уметь хранить их во внешней памяти. Реализация этой возможности с нуля означала бы, что сроки реализации Семантического Web были бы отодвинуты надолго. Тем более, уже существует зрелая технологиия эффективного хранения больших объемов данных во внешней памяти и доступа к ним, применяемой в различных областях, – реляционные базы данных. Поэтому РСУБД стали использоваться в различных средствах, поддерживающих работу с основными форматами Семантического Web.

Появляющиеся разработки использовали возможности баз данных различным образом. Формализмом лежащим в основе языка OWL являются дескриптивные логики [10]; как показали исследования [11, 12], этот формализм во многом противоречит допущениям, применяемым в базах данных. Поэтому в ряде семантических средств (например, Jena и Sesame [13]) РСУБД использовались лишь как один из альтернативных способов хранения RDF-данных наряду с хранением в файлах или в основной памяти. В таких средствах логический вывод выполнялся с использованием стандартных систем рассуждений, оперирующих в основной памяти. Более того, в разработке Sesame было принято сознательное решение отказаться от использования методик оптимизации запросов в РСУБД, чтобы результат не зависел от выбранного метода хранения.

Приложение instanceStore [14] было одним из первых, нацеленных на решение проблем масштабируемости, оно использовало рассуждатели для построения иерархии концепций и базы данных для определения индивидуумов, принадлежащих некоторым концепциям. Однако приложение не поддерживало роли (бинарные отношения между ресурсами), а поддерживаемые запросы являются очень ограниченным подмножеством языка запросов SPARQL.

Использование средств, нацеленных на соблюдение свойств полноты логического вывода для OWL, не позволяло достигать приемлемой для практических приложений производительности и масштабируемости, что заставило исследователей заняться исследованием альтернативных способов рассуждений [15]. Отказ от полноты для достижения эффективности вычислений означал, что техники и возможности баз данных могут в большей степени использоваться в логическом выводе.

Были исследованы теоретические основы альтернативных способов рассуждений. В частности, в [16] было определено подмножество языка OWL, которое может быть отображено в логические правила с соблюдением свойства полноты. Стали появляться и разработки, в которых базы данных использовались для выполнения рассуждений на основе некоторого подмножества OWL (например, DLDB [17]).

2.1 Интегрированные подходы

Большинство предложений для поддержки форматов Семантического Web основывались на архитектуре “система рассуждений / база данных”. Это решение ограничивало использование техник оптимизации баз данных и связано с накладными расходами, связанными с необходимостью преобразования из форматов SQL в объекты языков программирования. Также важным недостатком является то, что пользователи базы данных не могут выполнять доступ к RDF-данным. Эти недостатки позволяют преодолеть интегрированные решения. В [18] предлагается использовать XML-средства СУБД IBM DB2 для поддержки транзитивных отношений в RDF. В Oracle также поддерживается хранение и доступ к RDF-данным и выполнение рассуждений для специально определенного подмножества OWL [19].

Автором на основе возможностей РСУБД, описанных в [8], разработаны и реализованы в СУБД PostgreSQL методы выполнения доступа к RDF-данным с учетом их семантики. Подробное описание данной работы приведено в [6]. В данной работе был реализован упрощенный движок SPARQL-запросов, поддерживающий только конъюнкцию шаблонов троек. Как отмечается в [6], более сложные конструкции SPARQL могут быть реализованы и с помощью такого упрощенного движка и операторов SQL. Однако такой подход требует от пользователя знания методов отображения конструкций SPARQL в SQL, связанных с рядом “подводных камней” [20]. Непосредственная поддержка данных конструкций также позволяет выполнять запросы более эффективно за счет использования техник оптимизации и сокращения числа кортежей,

передаваемых через интерфейс хранимой процедуры, реализующей движок запросов. Поэтому движок запросов [6] был улучшен для поддержки большинства конструкций SPARQL.

В [6] также делалось ограничение на шаблоны троек, допуская только те, в которых на позиции свойства указан ресурс, а для свойства `rdf:type` и на позиции объекта – ресурс. Подобное ограничение встречается и в других системах, включая Pellet. Использование стандартных способов рассуждений дескриптивных логик и альтернативных таких, как в [16], делает такие ограничения обязательными. Представление описаний в реляционных таблицах [21] и выполнение логического вывода с помощью SQL-запросов над этими таблицами позволил отказаться от данного ограничения в более полном движке запросов. Оно накладывалось в [6], так как запросы с шаблонами, нарушающими данное ограничение, могут быть достаточно "дорогостоящими", приводя к логическому выводу большого числа RDF-троек.

3 Представляя реляционные данные в RDF

Более полная реализация SPARQL привносит новые возможности, не свойственные SQL. Сравнивая реляционную модель данных и модель данных Семантического Web, обычно в соответствие таблицам ставят классы OWL, а их атрибутам - свойства. В обеих моделях одна и та же сущность (индивидуум) может содержаться в нескольких таблицах (классах). Очевидно, что в такой ситуации у пользователя может возникнуть потребность узнать, в каких таблицах хранится информация о сущности `I` (или каким классам принадлежит индивидуум `I`). Чтобы ответить на этот вопрос, используя SQL, пользователю придется сначала узнать, какие таблицы созданы в базе данных, и затем явно проверить для каждой из этих таблиц, содержит ли она искомую сущность. Использование SPARQL существенно упрощает эту задачу, поскольку для ее решения достаточно лишь одного запроса с шаблоном тройки `{I a ?C}`. Далее, для того, чтобы узнать значения всех атрибутов для `I` в SQL опять придется использовать запрос, в котором нужно явно указать все эти атрибуты и таблицы. Используя SPARQL, пользователь не обязан знать, какими именно свойствами обладает конкретный индивидуум, чтобы узнать о нем подробную информацию: ему достаточно использовать запрос с шаблоном `{I ?p ?v}`.

Эти возможности могут быть использованы и применительно к данным из реляционных таблиц, если последние будут каким-либо образом отображены в RDF. Заметим, что наличие такого отображение полезно и с точки зрения полноты результатов запросов к RDF-данным с учетом их семантики. В [6] приведен пример того, как запросы к RDF-данным могут использоваться в SQL-запросах к обычным таблицам. В этом примере в базу данных загружалась онтология, описывающая родственные отношения, и было показано, как RDF-данные позволяли определять родственников, работающих в одном отделе, при этом информация о работниках отделов хранилась в таблице. Таким образом, RDF-данные дополняют данные из обычных таблиц, позволяя получать ответы на более интересные запросы. Допустим, что в базе данных из этого примера помимо информации об отделах также содержится таблица с информацией о родителях и детях сотрудников. Представив эти отношения в RDF, мы смогли бы их использовать во время логического вывода на основе загруженной онтологии. Если в результате этого отображения были получены новые тройки, ранее не содержавшиеся в RDF-хранилище, то SPARQL-запросы о семейных отношениях будут возвращать большее число строк. В результате пользователь сможет узнать информацию о большем числе родственников, работающих в одном отделе. Тем самым, возможность представления реляционных данных в RDF сделает обмен информации между RDF-хранилищем и SQL двусторонним.

4 Метод отображения

Рассмотрим простейший метод представления данных реляционных таблиц в RDF, который может использоваться в интегрированных подходах как [6]. Мы будем предполагать, что в каждой таблице содержится информация об отдельной сущности. Таблицы будут отображаться в классы, атрибуты – в свойства, а строки – в наборы троек, в которых записаны значения свойств одного индивидуума.

Создадим в базе данных следующие хранимые процедуры:

- `mapAttToProp(tbl,att,rdfProp)` - декларировать, что атрибут `att` таблицы `tbl` должен отображаться в свойство `rdfProp`. Свойство `rdfProp` должно содержаться в словаре RDF-хранилища и для него должен быть определен ранг. Декларация считается ошибочной, если тип данных атрибута и ранг свойства несовместимы.
- `mapTblToCls(tbl,owlClass)` - указать, что таблица `tbl` будет отображена в класс `owlClass`, который также должен содержаться в словаре.
- `addKeyProp(tbl,rdfProp)` - указать, что свойство `rdfProp` является частью ключа отображения строк таблицы `tbl` в индивидуумы класса `owlClass`.
- `execMapping()` - выполнить отображения таблиц в классы на основе вышеуказанных деклараций. Для каждой группы строк добавленной для отображения таблицы, имеющей одинаковые значения атрибутов ключа, определенного с помощью `addKeyProp()`, в словаре создается новый ресурс. Таким образом, каждой строке таблицы в соответствие ставится один ресурс, но одному ресурсу может соответствовать несколько строк одной таблицы. Затем по каждой строке таблицы устанавливаются значения атрибутов соответствующего ресурса на основе деклараций `mapAttToProp()`. Поскольку отображение строк на ресурсы не взаимнооднозначно, свойства могут иметь множественные значения.

Заметим, что ресурсы могут не создаваться заново для тех групп строк, для которых в RDF-хранилище уже существует некоторый индивидуум `I`, у которого установлены значения всех свойств, входящих в ключ отображения, и эти значения равны соответствующим значениям атрибутов данных строк. В этом случае остальные атрибуты строк будут отображаться в свойства индивидуума `I`. Такое поведение позволит сократить размер словаря.

Представленный метод отображения является упрощенным в том смысле, что позволяет создавать только тройки со свойствами, значения которых литералы. В то же время большинство свойств в сегодняшних онтологиях являются объектными, то есть имеют значения-ресурсы. Эта проблема может быть решена без изменения метода путем добавления в загруженную онтологию новых SWRL-правил.

Поясним на примере: пусть в базе данных определена таблица `person(emp_id int, full_name varchar(18), sex bool, parent_id int)`. С помощью вызовов хранимых процедур `mapAttToProp(person,emp_id,ex:id)`, `mapAttToProp(person,full_name,ex:name)`, `mapAttToProp(person,sex,intnl:sex_id)` и `mapAttToProp(person,parent_id,intnl:parent_id)` указывается соответствие между атрибутами и свойствами. Вызов `addKeyProp(person,ex:id)` указывает, что свойство `ex:id` ключевым при отображении таблицы `person`, а вызов `mapTblToCls(person,ex:Person)` указывает, что таблица отображается в класс `ex:Person`. Заметим, что в результате такого отображения свойство `intnl:parent_id` может иметь несколько значений, что естественно.

Далее, пусть в результате такого отображения в RDF-хранилище содержатся тройки `<db:per1 intnl:parent_id 28>` и `<db:per28 ex:id 28>`. Эти две тройки сами по себе не связывают ресурсы `db:per1` и `db:per28` каким-либо отношением. Однако достаточно добавить в онтологию следующее SWRL-правило:

```
indivPropertyAtom(?x,ex:hasParent,?y) :- classAtom(?x,ex:Person),
                                         classAtom(?y,ex:Person),
                                         dataPropertyAtom(?x,intnl:parent_id,?z),
                                         dataPropertyAtom(?y,ex:id,?z)
```

для того, чтобы отношение было задано неявно. Пусть, кроме того, тройка `<db:per28 intnl:sex_id 'true'>` должна определять, что ресурс `db:per28` соответствует мужчине. Однако, в онтологии мужской пол человека записывается с помощью объектного свойства `ex:hasSex` или класса `ex:Man`. Для того, чтобы записать информацию о поле в соответствии с онтологией, добавляем в нее правило:

```
indivPropertyAtom(?x,ex:hasSex,ex:Male) :- dataPropertyAtom(?x,intnl:sex_id,'true')
```

Альтернативным способом является использование OWL-описания:

$$\exists \texttt{intnl:sex\_id}.\{\texttt{'true'}\} \sqsubset \texttt{ex:Man}$$

Значения объектных свойств в данном случае получаются на основе литеральных значений свойств, полученных при отображении. Однако последние могут быть не интересны сами по себе и лишь увеличивать размеры RDF-хранилища. Чтобы избежать этого избыточного хранения, в ряде случаев (как при указании родителей) полезным было бы сразу указывать, что значения некоторых столбцов должны использоваться для отображения в значение объектного свойства. Для этого может быть создана дополнительная хранимая процедура `addColToFK(tbl,att,cls,rdfProp)` для того, чтобы указать, что атрибут `att` таблицы `tbl` соответсвует свойству `rdfProp` класса `cls`. Вызов этой процедуры должен быть сделан для каждого свойства, входящего в ключ отображения некоторой таблицы в класс `cls`. Тогда эти декларации могут быть использованы для создания сразу объектных свойств при вызове процедуры `execMapping()`.

5 Заключение

В статье рассмотрены различные способы решения актуальной задачи использования СУБД для хранения и выполнения доступа к RDF-данным. Для интегрированных решений предлагается метод наполнения RDF-хранилищ из реляционных таблиц, позволяющих на основе онтологий OWL обнаруживать отношения между сущностями, информация о которых хранится в базе данных. Представление баз данных в RDF позволяет также выполнять к ним SPARQL-запросы, обладающие рядом интересных возможностей в сравнении с SQL. В результате использования отображения может достигаться тесная интеграция RDF-данных и данных из реляционных баз данных. Для проверки предложенного метода планируется его реализация в средстве [6].

В результате отображения на основе предложенного метода в RDF-хранилище, по сути, может быть создана копия базы данных, что связано с двумя проблемами: 1) поддержание двух версий (исходной базы данных и полученного при отображении набора RDF-троек) в согласованном состоянии; 2) накладные расходы, связанные с хранением “RDF-копии” базы данных. Указанные недостатки могут быть незначительными в случаях, когда в отображении участвуют редко модифицируемые таблицы или таблицы, над которыми выполняются только инкрементальные обновления. Преодоление этих проблем для часто модифицируемых таблиц может являться задачей для дальнейшего развития метода.

Список литературы

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. *The Semantic Web.* // Scientific American, 284(5), 2001. P. 35–43.

[2] Christoph Bussler. *Is Semantic Web Technology Taking the Wrong Turn?* // IEEE Internet Computing, 12(1), 2008. P. 75–79.

[3] Jim Hendler. *Web 3.0 Emerging.* // IEEE Computer, 42(1), 2009. P. 111–113.

[4] Дмитрий Левшин. *Web, часть третья.* // Открытые Системы, № 3, 2009. С. 50–53.

[5] James Geller, Soon Ae Chun, and Yoo Jung An. *Towards the Semantic Deep Web.* // IEEE Computer, 41(9), 2008. P. 95–97.

[6] Dmitry V. Levshin. *An SQL-based Approach to SemanticWeb Reasoning and Query Answering.* // Proc. of the Spring Young Researcher’s Colloquium on Database and Information Systems, Saint-Petersburg, Russia, 2009.

[7] W3C Semantic Web Activity. [HTML] (http://www.w3.org/2001/sw/).

[8] Дмитрий Левшин. *Об Интеграции Semantic Web с PostgreSQL.* // Сборник статей молодых ученых факультета ВМиК МГУ, Выпуск 4, 2007. С. 92–100.

[9] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. *Pellet: A Practical OWL DL Reasoner.* // Journal of Web Semantics, 5(2), 2007. P. 51–53.

[10] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[11] Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. *Can OWL and Logic Programming Live Together Happily Ever After?* // Proc. of the 5th International Semantic Web Conference (ISWC 2006), 2006. P. 501–514.

[12] Evren Sirin, Michael Smith and Evan Wallace. *Opening, Closing Worlds - On Integrity Constraints.* // OWLED, volume 432 of CEUR Workshop Proceedings, 2008.

[13] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. *Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema.* // International Semantic Web Conference, 2002. P. 54–68.

[14] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. *The Instance Store: DL Reasoning with Large Numbers of Individuals.* // Proc. of the 2004 Description Logic Workshop (DL 2004), 2004. P. 31–40.

[15] Raphael Volz. *Change Paths in Reasoning!* // Proc. of the 1st International Workshop "New Forms of Reasoning for the Semantic Web", volume 291 of CEUR Workshop Proceedings, 2007.

[16] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. *Description Logic Programs: Combining Logic Programs with Description Logic.* // WWW, 2003. P. 48–57.

[17] Zhengxiang Pan and Jeff Heflin. *DLDB: Extending Relational Databases to Support Semantic Web Queries.* // PSSS, 2003. P. 109–113.

[18] Lipyeow Lim, Haixun Wang, and Min Wang. *Semantic Data Management: Towards Querying Data with their Meaning.* // Proc. of the 23rd International Conference on Data Engineering (ICDE 2007), 2007. P. 1438–1442.

[19] Eugene Inseok Chong, Souripriya Das, George Eadon, and Jagannathan Srinivasan. *An Efficient SQL-based RDF Querying Scheme.* // VLDB, 2005. P. 1216–1227.

[20] Richard Cyganiak. *A relational algebra for SPARQL.* // Tech. Rep. HPL-2005-170, Digital Media Systems Laboratory, HP Laboratories Bristol, 2005.

[21] Дмитрий Левшин. *Метод представления описаний в web-онтологиях для реализации рассуждений в реляционных СУБД.* // Микроэлектроника и информатика – 2009. 16-я Всероссийская межвузовская научно-техническая конференция студентов и аспирантов: Тезисы докладов, 2009. М.: МИЭТ. С. 177.

УДК 517.927.25

О СКОРОСТИ СХОДИМОСТИ СПЕКТРАЛЬНЫХ РАЗЛОЖЕНИЙ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ ОПЕРАТОРОВ ВТОРОГО ПОРЯДКА.

© 2009 г. А. С. Марков

markov.cs@gmail.com

Кафедра Общей математики

1 Введение

Данная работа посвящена изучению свойств систем корневых функций линейных обыкновенных дифференциальных операторов L, заданных на конечном отрезке числовой прямой. Операторы могут быть как самосопряженными, так и несамосопряженными, причем особое внимание уделено случаю существенно несамосопряженных операторов, системы корневых функций которых содержат бесконечное число присоединенных функций.

Установлена равносходимость вплоть до границы отрезка биортогональных разложений функций по корневым функциям операторов L с разложением этой функции в обычный тригонометрический ряд Фурье. Это сделано для дифференциального оператора второго порядка, коэффициент при первой производной которого - негладкая функция, принадлежащая лишь классу $\mathcal{L}^s, s > 1$. Получены точные оценки скорости равносходимости на всем интервале. Показана зависимость скорости равносходимости от расстояния от компакта до границы интервала. Установлено, что полученные оценки скорости равносходимости могут существенно зависеть от степени суммируемости s коэффициента при первой производной.

2 Постановка задачи

Рассмотрим произвольный дифференциальный оператор L, порожденный дифференциальной операцией

$$lu \equiv u'' + p_1(x)u' + q_1(x)u, \qquad x \in G = (0,1),$$

на классе функций D, абсолютно непрерывных на $\overline{G} = [0,1]$ вместе со своей первой производной;

$$p_1(x) \in \mathcal{L}^s(G,\mathcal{C}), \qquad s > 1, \qquad q_1(x) \in \mathcal{L}(G,\mathcal{C}). \tag{1}$$

Дадим определение корневых функций (т.е. собственных и присоединенных функций) и приведем основные условия на операторы. Корневые функции оператора L определим в обобщенном (по В.А. Ильину) смысле, рассматривая их как регулярные решения дифференциальных уравнений и освобождая их от требования удовлетворения каким-либо конкретным краевым условиям. Ограничения при этом налагаются на свойства спектра и корневых функций оператора. Это позволяет изучать как системы функций типа системы экспонент, не удовлетворяющие никаким краевым условиям без спектрального параметра, так и корневые функции конкретных краевых задач. Рассмотрены два встречающихся типа спектральных задач (отличающиеся нормировкой присоединенных функций). Под собственной функцией оператора L, отвечающей собственному значению $\lambda^2 \in \mathcal{C}$, будем понимать любую не равную тождественному нулю функцию $\overset{0}{u}(x) \in D$, удовлетворяющую почти всюду в G уравнению $l\overset{0}{u} + \lambda^2\overset{0}{u} = 0$. Под присоединенной функцией порядка $m, m = 1,2,...$, отвечающей тому же λ^2 и собственной функции $\overset{0}{u}$, будем понимать любую функцию $\overset{m}{u}(x)$, которая почти всюду в G удовлетворяет уравнению $l\overset{m}{u} + \lambda^2\overset{m}{u} = \mu_m\overset{m-1}{u}$. Здесь либо $\mu_m = 1$ (спектральная задача 1), либо $\mu_m = \lambda(\mathbf{Re}\lambda \geqslant 0)$ при

$|\lambda| \geqslant 1$, $\mu_m = 1$ при $|\lambda| < 1$ (спектральная задача 2). Приведем основные ограничения на рассматриваемые системы корневых функций. В случае конкретных краевых задач эти условия достаточно легко проверяются.

Фиксируем произвольную систему собственных значений $\{\lambda_k^2\}_{k=1}^{\infty}$ и произвольную систему $u_k(x)$ корневых функций оператора L, отвечающую этим собственным значениям, удовлетворяющие следующим трем *условиям В.А. Ильина,* назовем их *условиями I:*

1) система $\{u_k(x)\}$ замкнута и минимальна в $\mathcal{L}^r(G)$ при некотором $r \in [1, \infty]$;

2) существуют $c_1, c_2 = const > 0$ такие, что

$$|\mathbf{Im}\lambda_k| \leqslant c_1 \quad \forall k; \qquad \sum_{0 \leqslant |\lambda_k| - \lambda \leqslant 1} 1 \leqslant c_2 \quad \forall \lambda \geqslant 0; \tag{2}$$

3) существует $c_3 = const > 0$ такая, что

$$||u_k||_r ||v_k||_{r'} \leqslant c_3 \quad \forall k, \tag{3}$$

где $\{v_k\}-$ биортогонально сопряженная с $\{u_k\}$ система функций:
$v_k \in \mathcal{L}^{r'}(G)$, $(u_k, v_j) = \delta_{kj}$ $\forall k, j \in \mathcal{N}$, $r' = r/(r-1)$; $||\cdot||_r-$ обозначение нормы в $\mathcal{L}^r(G)$.

Будем также обозначать через $||f||_{r,E}$ норму функции $f(x) \in \mathcal{L}^r(E)$, $r \in [1, \infty]$.

Для произвольной функции $f(x) \in \mathcal{L}^r(G)$ составим частичные суммы биортогонального разложения

$$\sigma_\lambda(x, f) = \sum_{|\lambda_k| \leqslant \lambda} f_k u_k(x), \quad \lambda > 0, \quad f_k \equiv (f, v_k).$$

Через $S_\lambda(x, f)$ обозначим частичную сумму тригонометрического ряда Фурье функции $f(x)$, рассматриваемого как ортогональное разложение $f(x)$ для оператора $L_0 u = u''$ с условиями периодичности в 0 и 1.

Наложим дополнительно ограничение на систему $\{u_k, v_k\}$ и функции $f(x)$:

$$\exists \nu, \beta = const > 0 : \qquad \alpha_k f_k = O(\lambda_k^{-\nu} \ln^{-\beta} |\lambda_k|), \quad |\lambda_k| > 1, \tag{4}$$

$$\Big(\exists \nu = const > 0 : \qquad \alpha_k f_k = O(\lambda_k^{-\nu}), \tag{4'}\Big)$$

где $\alpha_k = ||v_k||_{r'}^{-1}$. Предположим далее, что для оператора L_0 условие (4) ((4′)) выполняется.

Основная задача состоит в том, чтобы установить факт равносходимости спектральных разложений $\sigma_\lambda(x, f)$ и $S_\lambda(x, f)$ функции $f(x)$ в метрике пространств $\mathcal{L}^p$, $p \geqslant 1$, на всём интервале и оценить скорость равносходимости указанных разложений (или, другими словами, оценить погрешность аппроксимации одного разложения другим). Из этих оценок будет следовать, в частности, что оба разложения сходятся или расходятся в метрике данного пространства $\mathcal{L}^p$ одновременно. Требуется при этом выявить характер зависимости оценок скорости равносходимости от показателя s суммируемости коэффициента $p_1(x)$ при первой производной дифференциального оператора L и от расстояния от компакта до границы основного интервала.

3 Равносходимость на всем интервале (0,1)

Теорема 1. *Пусть для оператора L и функции $f(x)$ выполняются условия (1),(4) и условия I. Тогда для всех достаточно больших чисел λ справедлива оценка*

$$\Big|\Big|\sigma_\lambda(x, f) - S_\lambda(x, f)\Big|\Big|_{\mathcal{L}^p(G)} \leqslant O\left(\max\left(\frac{1}{\lambda}, \frac{1}{\lambda^{\nu+\frac{1}{p}-1}}, \frac{n_1 \ln^2 \lambda}{\lambda^\nu \ln^\beta \lambda}\right)\right) \tag{5}$$

с постоянной c, не зависящей от λ; $n_1 = 1$, если присутствуют присоединенные функции и $||p_1||_1 \neq 0$, и $n_1 = 0$ в противном случае.

Доказательство.
Доказательство следует статьям И.С. Ломова [7], [10].
Обозначим $\sum^{\lambda} = \sum_{|\lambda_k| \leqslant \lambda}$, $\Theta_\lambda(x,y) = \sum^{\lambda} u_k(x)\bar{v}_k(y)$. Тогда $\sigma_\lambda(x,f) = \int\limits_{\overline{G}} \Theta_\lambda(x,y)f(y)\,dy$.
Фиксируем любой отрезок $K_1 = [a,b] \subset G$. Справедливо представление

$$||\sigma_\lambda(x,f) - S_\lambda(x,f)||_p^p = ||\cdot||_{p,(0,a)}^p + ||\cdot||_{p,K_1}^p + ||\cdot||_{p,(b,1)}^p \equiv \Delta_\lambda^a + \Delta_\lambda + \Delta_\lambda^b. \tag{6}$$

Оценим слагаемое

$$\Delta_\lambda \equiv \int\limits_K |\sigma_\lambda(x,f) - S_\lambda(x,f)|^p dx = \int\limits_K \left| \int\limits_{\overline{G}} f(y)[\Theta_\lambda(x,y) - D_\lambda(x,y)]dy \right|^p dx. \tag{7}$$

Зафиксировав произвольное число $R_0 \in (0, (1/2)\mathrm{dist}(K, \partial G))$, для любого числа $R \in [R_0/2, R_0]$ и любого $x \in K$ рассмотрим вспомогательную функцию

$$\omega_\lambda(\rho, R) = \begin{cases} (\sin\lambda\rho)/(\pi\rho), & |\rho| \leqslant R, \rho = x - y, \\ 0, & |\rho| > R. \end{cases} \tag{8}$$

Далее, как и в работах В.А. Ильина [3], [4], усредним функцию (8) по R :

$$S_0[\omega_\lambda(\rho,R)] = \frac{8}{3R_0^2} \int\limits_{R_0/2}^{R_0} R\omega_\lambda(\rho,R)dR \quad (S_0[1] = 1), \tag{9}$$

добавим и вычтем под знаком модуля в правой части (7) выражение (9) и оценим модуль суммы через сумму модулей ($c_p = 2^{p-1}$, $v_0(x,y) = S_0[\omega_\lambda(\rho,R)]$) :

$$\Delta_\lambda \leqslant c_p \int\limits_K \left| \int\limits_{\overline{G}} f(y)[\Theta_\lambda(x,y) - v_0(x,y)]dy \right|^p dx + c_p \int\limits_K \left| \int\limits_{\overline{G}} f(y)[D_\lambda(x,y) - v_0(x,y)]dy \right|^p dx \equiv$$

$$\equiv c_p(I + \widehat{I}). \tag{10}$$

Оценим первый интеграл I в правой части (10); интеграл $\widehat{I}$ оценивается аналогично.
Заменим функцию $v_0(x,y)$ биортогональным рядом. При выполнении условия (1) и условий I справедливо равенство $\forall p \in [1,+\infty)$, $\forall f(x) \in \mathcal{L}^r(G)$ [8]:

$$\lim_{\lambda\to\infty} \left\| \int\limits_G f(y)\Big[v_0(x,y) - \sum_{|\lambda_k|\leqslant\lambda} (v_0, \bar{u}_k)(x)\bar{v}_k(y)\Big]dy \right\|_{p,k} = 0. \tag{11}$$

Равенство (11) может быть верно и в равномерной метрике. Действительно, пусть для некоторого $\alpha \in [1,\infty)$

$$f(y) \overset{\mathcal{L}^\alpha(K_0)}{=} \sum_k f_k u_k(y), \quad K_0 = [a - R_0, b + R_0]. \tag{12}$$

Здесь и далее обозначено через $\sum\limits_k$ сумма $\sum\limits_{k=1}^{\infty}$. Умножим обе части (12) скалярно на функцию $S_0[w_\lambda(\rho,R)]$. Так как ряд в (12) сходится в $\mathcal{L}^\alpha(K_0)$, то на K_0 его можно почленно интегрировать, а вне K_0 функция $S_0[w_\lambda(\rho,R)]$ как функция y при $x \in K$ равна нулю. Получаем равенство

$$\int\limits_G f(y)v_0(x,y)dy = \sum_k f_k \int\limits_G u_k(y)v_0(x,y)dy \quad \forall x \in K. \tag{13}$$

Для функций u_k имеет место формула среднего значения (легко получаемая интегрированием по частям после замены A_1u_k на $u_k'' + \lambda_k^2 u_k$). Обозначим

$$s_t(u_k(x)) = u_k(x+t) + u_k(x-t). \tag{14}$$

Тогда $\forall \lambda_k \in \mathcal{C}, \quad x \pm t \in G$:

$$s_t(u_k(x)) = 2u_k(x)\cos\lambda_k t - h_k(x,t), \tag{15}$$

где

$$h_k(x,t) = \frac{1}{\lambda_k}\int\limits_{x-t}^{x+t} A_1u_k(\tau)\sin\lambda_k(|x-\tau|-t)d\tau, \quad \left.\frac{\sin\lambda_k(|x-\tau|-t)}{\lambda_k}\right|_{\lambda_k=0} = |x-\tau|-t,$$

а $A_1u_k \equiv -p_1u_k' - q_1u_k$, если u_k- собственная функция, $A_1u_k \equiv A_1\overset{m}{u}_k = \mu\overset{m-1}{u}_k - p_1\overset{m}{u}_k' - q_1\overset{m}{u}_k$, если $u_k = \overset{m}{u}_k-$ m-я присоединенная функция.

Исследуем в правых частях (11), (13) структуру коэффициентов

$$\int\limits_{\overline{G}} S_0[\omega_\lambda(\rho,R)]u_k(y)dy = \frac{1}{\pi}S_0\left[\int\limits_{x-R}^{x+R}\frac{\sin\lambda(x-y)}{x-y}u_k(y)dy\right] = \frac{1}{\pi}S_0\left[\int\limits_0^R\frac{\sin\lambda t}{t}s_t(u_k(x))dt\right] =$$

$$= \frac{1}{\pi}S_0\left[2u_k(x)\int\limits_0^R\frac{\sin\lambda t\cos\lambda_k t}{t}dt - \int\limits_0^R\frac{\sin\lambda t}{t}h_k(x,t)dt\right] =$$

$$= u_k(x)\delta_k^\lambda + u_k(x)I_k^\lambda(R_0) - \frac{1}{\pi}S_0\left[\int\limits_0^R\frac{\sin\lambda t}{t}h_k(x,t)dt\right]. \tag{16}$$

Здесь в проведенных преобразованиях мы сделали замены переменных $y-x=t$ при $y \geqslant x$ и $x-y=t$ при $y<x$, воспользовались формулой среднего (15) и леммой о разрывном множителе Дирихле в $\mathcal{C}$ [5]. По этой лемме справедливы соотношения

$$\frac{2}{\pi}S_0\left[\int\limits_0^R\frac{\sin\lambda t\cos\lambda_k t}{t}dt\right] = \delta_k^\lambda + I_k^\lambda(R_0),$$

$$\delta_k^\lambda = \begin{cases}1, & |\lambda_k| \leqslant \lambda,\\ 0, & |\lambda_k| > \lambda,\end{cases} \qquad I_k^\lambda(R_0) = \begin{cases}O(1), & \forall\lambda_k \in \mathcal{C},\\ O(|\lambda-|\lambda_k||^{-2}), & |\lambda_k| \geqslant 1, |\lambda-|\lambda_k|| \geqslant 1.\end{cases} \tag{17}$$

Подставим правую часть (16) в (11), (13) и соответственно в первый интеграл I в (10). Учитывая структуру δ_k^λ из (17), для интеграла I получаем выражение

$$I = \int_K \left|\sum_k f_k u_k(x) I_k^\lambda(R_0) - \frac{1}{\pi}S_0\Big[\int\limits_0^R\frac{\sin\lambda t}{t}h_k(x,t)dt\Big]\right|^p dx. \tag{18}$$

Для исследования выражения (18) воспользуемся леммой об оценках интегралов из работы И.С. Ломова [9]. Фиксируем произвольно числа $\alpha, \beta \in \overline{G}$, $\alpha < \beta$, $R \in [R_0/2, R_0]$,

$\tau \in [0,R]$, $s \in \mathcal{R}$, $s>1$, $\lambda > 0-$ достаточно большое число $(\lambda > 4)$, $\{\lambda_k\} \in \mathcal{C}$, $|\mathbf{Im}\lambda_k| \leqslant c = const < \infty$, $k = 1,2,...$ Рассмотрим следующие интегралы:

$$K_0(\lambda,\lambda_k,\tau,R) \equiv \frac{1}{\lambda_k}\int\limits_{\tau}^{R} \frac{\sin\lambda r \sin\lambda_k(r-\tau)}{r}dr, \quad \lambda_k \neq 0,$$

$$A_k^s(\lambda,\alpha,\beta,R) \equiv \left(\int\limits_{\alpha}^{\beta} |K_0(\lambda,\lambda_k,\tau,R)|^s d\tau\right)^{1/s}, \quad A_k \equiv A_k^1(\lambda,\alpha,\beta,R).$$

Лемма 1 *Равномерно по $\alpha,\beta,R,\lambda,\lambda_k$ из указанных выше множеств имеют место следующие оценки:*

1) $I_k^\lambda = O(\lambda^{-2})$, $A_k = O(\lambda^{-1})$, $A_k^s = O(\lambda^{-1})$ *при* $|\lambda_k| \leqslant 1$;

2) $I_k^\lambda = O(\lambda^{-2})$, $A_k = O((\lambda|\lambda_k|)^{-1}\ln|\lambda_k|)$, $A_k^s = O(\lambda^{-1}|\lambda_k|^{-\frac{1}{s}})$ *при* $1 \leqslant |\lambda_k| \leqslant \frac{\lambda}{2}$;

3) $I_k^\lambda = O((\lambda|\lambda_k|)^{-1})$, $A_k = O(|\lambda_k|^{-2}\ln\lambda)$, $A_k^s = O(\lambda^{1-\frac{1}{s}}|\lambda_k|^{-2})$ *при* $|\lambda_k| \geqslant \frac{3\lambda}{2}$;

4) $I_k^\lambda = O(1)$, $A_k = O(\lambda^{-1}\ln\lambda)$, $A_k^s = O(\lambda^{-\frac{1}{s}})$ *при* $|\lambda - |\lambda_k|| \leqslant \frac{2}{R_0}$;

5) $I_k^\lambda = O(|\lambda-|\lambda_k||^{-2})$, $A_k = O((|\lambda_k||\lambda-|\lambda_k||)^{-1}\ln\lambda)$, $A_k^s = O(\lambda^{1-\frac{1}{s}}(|\lambda_k||\lambda-|\lambda_k||)^{-1})$ *при* $\frac{2}{R_0} \leqslant |\lambda-|\lambda_k|| \leqslant \frac{\lambda}{2}$.

В соответствии с произведенным в лемме разбиением множества $\{\lambda_k\}$ введем обозначения для сумм по рассматриваемым множествам значений λ_k: $\sum_1$ – сумма по тем λ_k, для которых справедливо неравенство $|\lambda_k| < 1$; $\sum_2 : \ 1 < |\lambda_k| \leqslant \lambda/2$; $\sum_3 : \ |\lambda_k| \geqslant 3\lambda/2$; $\sum_4 : \ |\lambda - |\lambda_k|| \leqslant 2/R_0$; $\sum_5 : \ 2/R_0 \leqslant |\lambda-|\lambda_k|| \leqslant \lambda/2$.

Рассмотрим выражение в правой части (18), содержащее $I_k^\lambda(R_0)$. Воспользовавшись для I_k^λ оценками из леммы и применив неравенство [7]

$$\exists c = const : \quad ||u_k||_\infty \alpha_k^{-1} \leqslant c \quad \forall k \in \mathcal{N}, \tag{19}$$

для достаточно больших λ получаем

$$\sum_k |f_k u_k(x) I_k^\lambda(R_0)| \leqslant c\Big[\lambda^{-2}\sum\nolimits_1 |\hat{f}_k| + \lambda^{-2}\sum\nolimits_2 |\hat{f}_k| + \lambda^{-1}\sum\nolimits_3 |\hat{f}_k\lambda_k^{-1}| + \sum\nolimits_4 |\hat{f}_k| +$$

$$+ \sum\nolimits_5 |\hat{f}_k||\lambda-|\lambda_k||^{-2}\Big], \quad \hat{f}_k \equiv \alpha_k f_k, \tag{20}$$

равномерно по $x \in G$ (обозначение α_k введено в условии (4)).

Оценим суммы в правой части последнего неравенства. В сумме $\sum_1$ используем неравенство Гельдера для $f_k = (f,v_k)$ и второе условие из (2), получаем $\lambda^{-2}\sum_1 |\hat{f}_k| \leqslant c_2||f||_r\lambda^{-2}$. Для остальных сумм используем оценку (4) и также второе условие из (2). Сумму $\sum_2$ оценим следующим образом: $\lambda^{-2}\sum_2 |\hat{f}_k| \leqslant c\lambda^{-2}\sum\limits_{n=2}^{[\lambda/2]} n^{-\nu}\ln^{-\beta} n = \varphi(\lambda)$, где $\varphi(\lambda) = O(\lambda^{-2})$ при $\nu > 1$, $\varphi(\lambda) = O(\lambda^{-2}\ln^{-\beta+1}\lambda)$ при $\nu = 1$ и $\varphi(\lambda) = O(\lambda^{-\nu-1})$ при $\nu < 1$. Для суммы $\sum_3$ получаем $\lambda^{-1}\sum_3 |\hat{f}_k\lambda_k^{-1}| \leqslant c\lambda^{-1}\sum\limits_{n\geqslant[3\lambda/2]} n^{-\nu-1}\ln^{-\beta} n = O(\lambda^{-\nu-1}\ln^{-\beta}\lambda)$. Сумма $\sum_4$ имеет, очевидно,

оценку $O(\lambda^{-\nu}\ln^{-\beta}\lambda)$, а сумма $\sum_5 |\hat{f}_k||\lambda-|\lambda_k||^{-2} \leqslant O(\lambda^{-\nu}\ln^{-\beta}\lambda)$. Объединяя установленные оценки, получаем равномерно по $x\in\overline{G}$

$$\sum_k |f_k u_k(x) I_k^\lambda(R_0)| = O(\max(\frac{1}{\lambda^2},\ \frac{1}{\lambda^\nu \ln^\beta \lambda})). \tag{21}$$

Будем считать далее, не ограничивая общности, что $\lambda_k \neq 0$ (если $0\in\{\lambda_k\}$, то в силу второго условия (2) можем изменить $q_1(x)$ так, что $0\notin\{\lambda_k\}$).

Преобразуем оставшуюся часть под знаком интеграла в (18). Сделаем замены переменных $x-r=\tau$ для $x>r$ и $r-x=\tau$ для $x<r$, а затем поменяем местами интегралы:

$$S(x) \equiv -\frac{1}{\pi}\sum_k \frac{f_k}{\lambda_k} S_0\left[\int_0^R \frac{\sin\lambda t}{t}\int_{x-t}^{x+t} A_1 u_k(r)\sin\lambda_k(|x-r|-t)\,dr dt\right] =$$

$$= -\frac{1}{\pi}\sum_k \frac{f_k}{\lambda_k} S_0\left[\int_0^R \frac{\sin\lambda t}{t}\int_0^t s_\tau(A_1u_k(x))\sin\lambda_k(\tau-t)\,d\tau dt\right] =$$

$$= \frac{1}{\pi}\sum_k f_k S_0\left[\int_0^R K_0(\lambda,\lambda_k,\tau,R) s_\tau(A_1u_k(x))\,d\tau\right]. \tag{22}$$

Обозначение $s_\tau(\cdot)$ введено в (14).

Расщепим в (22) A_1u_k на три слагаемых в соответствии с обозначением после формулы (15) и запишем (22) в виде $S=S_1+S_2+S_3$, где $S_j(x)$, $j=1,2,3$, получается из $S(x)$ в (21) после замены A_1u_k на $(-q_1u_k)$, $\mu_k \overset{m-1}{u_k}$ и $(-p_1u_k')$ соответственно (считаем, что $u_k=\overset{m}{u_k}-$ присоединенная функция). Применяя обобщенное неравенство Минковского [2], покажем, что операция усреднения (9) не влияет на последующие преобразования:

$$||S_0[K_0]||_p \leqslant S_0[||K_0||_p] = S_0[A_k^p(\lambda,0,1,R)] \quad \forall p\geqslant 1. \tag{23}$$

Поскольку оценки леммы для интегралов A_k^s, A_k равномерны по R, а операция усреднения введена так, что $S_0[1]=1$, то на последующие оценки эта операция не оказывает влияния. Указанная операция существенно использована лишь при рассмотрении (17).

Для получения оценки суммы $S_1(x)$ применим неравенство Юнга [1], соотношения (19), (23) и оценки из леммы для A_k^p (используем обозначение $||\cdot||_{p,K}$ для нормы в $\mathcal{L}^p(k)$):

$$||S_1||_{p,K} \leqslant c||q_1||_1 \sum_k |\hat{f}_k| S_0[||K_0||_{p,(0,R_0)}] \leqslant c||q_1||_1\sum_k |\hat{f}_k| S_0[A_k^p(\lambda,0,R_0,R)] \leqslant$$

$$\leqslant c_0\left(\lambda^{-1}\sum\nolimits_1 |\hat{f}_k| + \lambda^{-1}\sum\nolimits_2 |\hat{f}_k\lambda_k^{-\frac{1}{p}}| + \lambda^{\frac{1}{q}}\sum\nolimits_3 |\hat{f}_k\lambda_k^{-2}| + \lambda^{-\frac{1}{p}}\sum\nolimits_4 |\hat{f}_k| + \lambda^{\frac{1}{q}}\sum\nolimits_5 |\hat{f}_k\lambda_k^{-1}||\lambda-|\lambda_k||^{-1}\right), \tag{24}$$

$$p\neq 1,\quad p^{-1}+q^{-1}=1.$$

Установим общую оценку для правой части (24) при условиях (2), (4). Как и выше, получаем $\lambda^{-1}\sum_1 |\hat{f}_k| = O(\lambda^{-1})$. Далее имеем $\lambda^{-1}\sum_2 |\hat{f}_k\lambda_k^{-1/p}| \leqslant c\lambda^{-1}\sum\limits_{n=2}^{[\lambda/2]} n^{-\nu-1/p}\ln^{-\beta} n = \varphi(\lambda)$, где $\varphi(\lambda)=O(\lambda^{-1})$ при $\nu>1/q$, $\varphi(\lambda)=O(\lambda^{-1}\ln^{-\beta+1}\lambda)$ при $\nu=1/q$ и $\varphi(\lambda)=O(\lambda^{-\nu-1/p})$ при $\nu<1/q$. Для суммы $\sum_3$ находим, что

$$\lambda^{1/q}\sum\nolimits_3 |\hat{f}_k\lambda_k^{-2}| \leqslant c\lambda^{1/q}\sum_{n\geqslant 3\lambda/2} n^{-\nu-2}\ln^{-\beta} n = O(\lambda^{-\nu-1/p}\ln^{-\beta}\lambda).$$

Оценка для суммы $\sum_4$ имеет вид $\lambda^{-1/q}\sum_4|\hat{f}_k| = O(\lambda^{-\nu-1/p}\ln^{-\beta}\lambda)$. Наконец, оцениваем сумму $\sum_5$:

$$\lambda^{1/q}\sum\nolimits_5|\hat{f}_k\lambda_k^{-1}||\lambda-|\lambda_k||^{-1} \leqslant c\lambda^{-\nu+1/q-1}\ln^{-\beta}\lambda\sum\nolimits_5|\lambda-|\lambda_k||^{-1} = O(\lambda^{-\nu-1/p}\ln^{-\beta+1}\lambda).$$

При $p=1$ преобразуем S_1 по той же схеме:

$$||S_1||_{p,K} \leqslant c||q_1||_1\sum_k|\hat{f}_k|S_0[A_k^1(\lambda,0,R_0,R)] \leqslant c_0\Bigg(\lambda^{-1}\sum\nolimits_1|\hat{f}_k| + \lambda^{-1}\sum\nolimits_2|\hat{f}_k\lambda_k^{-1}|\ln|\lambda_k| +$$

$$+\ln\lambda\sum\nolimits_3|\hat{f}_k\lambda_k^{-2}| + \lambda^{-1}\sum\nolimits_4|\hat{f}_k| + \ln\lambda\sum\nolimits_5|\hat{f}_k\lambda_k^{-1}||\lambda-|\lambda_k||^{-1}\Bigg) \leqslant c_2\lambda^{-1} \qquad \forall\nu>0,\ \forall\beta>0.$$

Объединяя установленные оценки, получаем

$$||S_1||_{p,K} = O(\max(\frac{1}{\lambda},\ \frac{\ln\lambda}{\lambda^{\nu+\frac{1}{p}}\ln^{\beta}\lambda},\ \frac{1}{\lambda^{\nu+\frac{1}{p}}})). \tag{25}$$

Рассмотрим выражение из $S_2(x)$, полученное применением формулы среднего (15):

$$I_\lambda \equiv \frac{1}{\lambda_k}S_0\Bigg[\int_0^R\frac{\sin\lambda t}{t}\int_0^t s_\tau(\mu_m\overset{m-1}{u_k}(x))\sin\lambda_k(t-\tau)\,d\tau dt\Bigg]. \tag{26}$$

Как и в работе И.С. Ломова [9], получим модифицированную формулу среднего ($x\pm t\in\overline{G}$, $\lambda_k\in\mathcal{C}$)

$$s_t(\overset{m}{u_k}(x)) = 2\sum_{l=0}^{m}\mu_m^l\overset{m-l}{u_k}(x)B_t^l(\cos\lambda_k t_l) - \sum_{l=0}^{m}\mu_m^l B_t^{l+1}[s_{t_{l+1}}(p_1(x)\overset{m-l}{u_k'}(x) + q_1(x)\overset{m-l}{u_k}(x))], \tag{27}$$

где $B_t^0(g(t_0)) \equiv g(t)$, $\quad B_t^1(g(t_1)) \equiv \frac{1}{\lambda_k}\int\limits_0^t g(t_1)\sin\lambda_k(t-t_1)\,dt_1$,

$$B_{t_0}^l(g(t_l)) \equiv \frac{1}{\lambda_k}\int_0^{t_0}\dots\int_0^{t_{l-1}}\Bigg[g(t_1)\prod_{j=1}^{l}\sin\lambda_k(t_{j-1}-t_j)\Bigg]dt_l\dots dt_1, \quad l=\overline{1,m_k};$$

при $\lambda_k=0$ следует положить $\lambda_k^{-1}\sin\lambda_k(t_{j-1}-t_j) \equiv t_{j-1}-t_j$. Формула для $s_\tau(\mu_m\overset{m-1}{u_k}(x))$ получается из (27) умножением на μ_m, заменой t на τ и m на $m-1$; обозначим $n=m-1$. Подставим полученное выражение в интеграл (26):

$$I_\lambda = \frac{2}{\lambda_k}\sum_{l=0}^{n}\mu_m^{l+1}\overset{n-l}{u_k}(x)S_0\Bigg[\int_0^R\frac{\sin\lambda t}{t}\int_0^t B_\tau^l(\cos\lambda_k t_l)\sin\lambda_k(t-\tau)\,d\tau dt\Bigg] -$$

$$-\frac{1}{\lambda_k}\sum_{l=0}^{n}\mu_m^{l+1}S_0\Bigg[\int_0^R\frac{\sin\lambda t}{t}\int_0^t B_\tau^{l+1}[s_{t_{l+1}}(p_1(x)\overset{n-l}{u_k'}(x) + q_1(x)\overset{n-l}{u_k}(x))]\sin\lambda_k(t-\tau)\,d\tau dt\Bigg] \equiv$$

$$\equiv I_{\lambda1}(x) - I_{\lambda2}(x), \tag{28}$$

где через $I_{\lambda1}$ обозначена первая сумма по l, через $I_{\lambda2}-$ вторая.

В $I_{\lambda 2}$ поменяем местами интегралы и воспользуемся для u_k оценками для производных и оценками антиприорного типа:

$$|I_{\lambda 2}| = \left| \sum_{l=0}^{n} \mu_m^{l+1} S_0 \left[\int_0^R K_0(\lambda, \lambda_k, \tau, R) B_\tau^{l+1} [s_{t_{l+1}}(p_1(x) \overset{n-l}{u_k'}(x) + q_1(x) \overset{n-l}{u_k}(x))] \, d\tau \right] \right| \leqslant$$

$$\leqslant c\alpha_k(|\lambda_k| \, ||p_1||_1 + ||q_1||_1) S_0[A_k^1(\lambda, 0, R_0, R)] \tag{29}$$

равномерно по $x \in K$. В сумме $I_{\lambda 1}$ рассмотрим интегралы

$$I_{\lambda 1}^l \equiv 2S_0 \left[\int_0^R \frac{\sin \lambda t}{t} \int_0^t B_\tau^l (\cos \lambda_k t_l) \sin \lambda_k (t - \tau) \, d\tau dt \right]. \tag{30}$$

Если $|\lambda_k| \leqslant 1$, то, проинтегрировав по t по частям ($\sin \lambda t$ интегрируем) и использовав первую из оценок (2), получаем

$$I_{\lambda 1}^l = O(\frac{1}{R_0 \lambda}), \quad |\lambda_k| \leqslant 1, \, \lambda > 1, \, l = \overline{0, n}. \tag{31}$$

Пусть $|\lambda_k| > 1$. Тогда

$$B_t^l (\cos \lambda_k t_l) = \frac{1}{2^{l+1}} \frac{1}{\lambda_k^l} [P_{1l}(t) \sin \lambda_k t + P_{2l}(t) \cos \lambda_k t], \tag{32}$$

где $P_{1l}(t)$−многочлен относительно t нечетной степени, $P_{1l}(-t) = -P_{1l}(t)$, с коэффициентами порядка $O(1)$ при $|\lambda_k| \geqslant 1$, $P_{1l}(0) = 0$; $P_{2l}(t)$−многочлен четной степени, $P_{2l}(-t) = P_{2l}(t)$, с коэффициентами порядка $O(1)$ при $|\lambda_k| \geqslant 1$, $P_{2l}(0) = 0$.

Поскольку интеграл по τ в (30) есть $\lambda_k B_t^{l+1} (\cos \lambda_k t_{l+1})$, то из (32) получаем

$$\int_0^t B_\tau^l (\cos \lambda_k t_l) \sin \lambda_k (t - \tau) \, d\tau = \frac{1}{2^{l+2}} \frac{1}{\lambda_k^l} [P_{1l+1}(t) \sin \lambda_k t + P_{2l}(t) \cos \lambda_k t].$$

Подставим это выражение в (30), сократим на t и преобразуем произведения тригонометрических функций в суммы. Тогда если $|\lambda - |\lambda_k|| \leqslant 1$, то

$$I_{\lambda 1}^l = O(\frac{1}{R_0 |\lambda_k|^l}), \quad |\lambda - |\lambda_k|| \leqslant 1, \, l = \overline{0, n}. \tag{33}$$

Если $|\lambda - |\lambda_k|| > 1$, то проинтегрируем (30) по t и учтем, что подстановки при $t = 0$ обратятся в нуль ($P_{2l+1}(t)/t = 0$ при $t = 0$ и либо $P_{1l+1}(t)/t = 0$ при $t = 0$, либо $\sin(\lambda \pm \lambda_k)t = 0$ при $t = 0$). В результате получаем сумму синусов и косинусов с аргументами $(\lambda \pm \lambda_k)R$ с коэффициентами, зависящими от R и имеющими порядок $O(\lambda_k^{-l} |\lambda - |\lambda_k||^{-1})$. Операция усреднения по R приводит к оценке

$$I_{\lambda 1}^l = O(\frac{1}{R_0 \lambda_k^l |\lambda - |\lambda_k||^2}), \quad |\lambda - |\lambda_k|| \geqslant 1, \, |\lambda_k| > 1, \, l = \overline{0, n}. \tag{34}$$

Таким образом, интегралы $I_{\lambda 1}^l$ удовлетворяют соотношениям (31), (33) и (34). Подставляя эти оценки в первую сумму (28), получаем при $|\lambda| > 1$

$$I_{\lambda 1} = \alpha_k O(\frac{1}{R_0 \lambda}), \, |\lambda_k| \leqslant 1; \quad I_{\lambda 1} = \alpha_k O(\frac{1}{R_0}), \, |\lambda - |\lambda_k|| \leqslant 1;$$

$$I_{\lambda 1} = \alpha_k O(\frac{1}{R_0 |\lambda - |\lambda_k||^2}), \, |\lambda - |\lambda_k|| > 1, \, |\lambda_k| > 1. \tag{35}$$

Оценки не изменятся, если сравнивать $|\lambda - |\lambda_k||$ с числом $2/R_0$ вместо 1. Этим завершено исследование интеграла I_λ (26).

Подставляя в сумму $S_2(x)$ оценки (29), (35) интеграла I_λ и используя лемму об оценках интегралов [9], получаем

$$|S_2(x)| \leqslant c\sum_k |\hat{f}_k|\big[I_{\lambda 1} + (|\lambda_k|\,||p_1||_1 + ||q_1||_1)S_0[A_k^1(\lambda,0,R_0,R)]\big] \leqslant$$

$$\leqslant \frac{c_0}{R_0}\left[\lambda^{-1}\sum\nolimits_1 |\hat{f}_k| + \lambda^{-2}\sum\nolimits_2 |\hat{f}_k| + \lambda^{-1}\sum\nolimits_3 |\hat{f}_k\lambda_k^{-1}| + \sum\nolimits_4 |\hat{f}_k| + \sum\nolimits_5 |\hat{f}_k||\lambda - |\lambda_k||^{-2}\right] +$$

$$+c_0\left[\lambda^{-1}\sum\nolimits_1 |\hat{f}_k| + \lambda^{-1}\sum\nolimits_2 (|\lambda_k|\,||p_1||_1 + ||q_1||_1)|\hat{f}_k\lambda_k^{-1}|\ln|\lambda_k| + \ln\lambda\sum\nolimits_3 (|\lambda_k|\,||p_1||_1 + ||q_1||_1)|\hat{f}_k\lambda_k^{-2}| +\right.$$

$$\left. +\lambda^{-1}\ln\lambda\sum\nolimits_4 (|\lambda_k|\,||p_1||_1 + ||q_1||_1)|\hat{f}_k| + \ln\lambda\sum\nolimits_5 (|\lambda_k|\,||p_1||_1 + ||q_1||_1)|\hat{f}_k\lambda_k^{-1}||\lambda - |\lambda_k||^{-1}\right]. \quad (36)$$

Параметр R_0 в данной теореме нас не интересует, поэтому положим в (36) $c_0/R_0 = C_0$.

Рассмотрим выражение, стоящее в первой квадратной скобке в правой части последнего неравенства, при условиях (2), (4). Для суммы $\lambda^{-1}\sum_1 |\hat{f}_k|$ получаем оценку $O(\lambda^{-1})$. Сумму $\sum_2$ оценим следующим образом: $\lambda^{-2}\sum_2 |\hat{f}_k| = \varphi(\lambda)$, где $\varphi(\lambda) = O(\lambda^{-2})$ при $\nu > 1$, $\varphi(\lambda) = O(\lambda^{-2}\ln^{-\beta+1}\lambda)$ при $\nu = 1$, $\varphi(\lambda) = O(\lambda^{-\nu-1})$ при $\nu < 1$. Сумма $\lambda^{-1}\sum_3 |\hat{f}_k\lambda_k^{-1}|$ имеет, очевидно, оценку $O(\lambda^{-\nu-1}\ln^{-\beta}\lambda)$. Суммы $\sum_4$ и $\sum_5$ имеют оценку $O(\lambda^{-\nu}\ln^{-\beta}\lambda)$. Таким образом, для первой квадратной скобки в (36) справедлива оценка

$$O(\max(\frac{1}{\lambda}, \frac{1}{\lambda^\nu \ln^\beta \lambda})). \quad (37)$$

Для оценки выражения, стоящего во второй квадратной скобке в правой части неравенства (36), воспользуемся условиями (2) и (4) и рассмотрим 2 случая:

I) $||p_1||_1 = 0$. В этом случае оценки сумм, стоящих во второй квадратной скобке рассматриваемого неравенства, выглядят следующим образом:

$$\lambda^{-1}\sum\nolimits_1 |\hat{f}_k| \leqslant O(\frac{1}{\lambda}); \quad \lambda^{-1}\sum\nolimits_2 ||q_1||_1|\hat{f}_k\lambda_k^{-1}|\ln|\lambda_k| = O(\frac{1}{\lambda});$$

$$\ln\lambda\sum\nolimits_3 ||q_1||_1|\hat{f}_k\lambda_k^{-2}| = O(\frac{\ln\lambda}{\lambda^{\nu+1}\ln^\beta\lambda}); \quad \lambda^{-1}\ln\lambda\sum\nolimits_4 ||q_1||_1|\hat{f}_k| = O(\frac{\ln\lambda}{\lambda^{\nu+1}\ln^\beta\lambda});$$

$$\ln\lambda\sum\nolimits_5 ||q_1||_1|\hat{f}_k\lambda_k^{-1}||\lambda - |\lambda_k||^{-1} = O(\frac{\ln^2\lambda}{\lambda^{\nu+1}\ln^\beta\lambda}).$$

Таким образом, при $||p_1||_1 = 0$ получаем следующую оценку:

$$\lambda^{-1}\sum\nolimits_1 |\hat{f}_k| + \lambda^{-1}\sum\nolimits_2 ||q_1||_1|\hat{f}_k\lambda_k^{-1}|\ln|\lambda_k| + \ln\lambda\sum\nolimits_3 ||q_1||_1|\hat{f}_k\lambda_k^{-2}| +$$

$$+\lambda^{-1}\ln\lambda\sum\nolimits_4 ||q_1||_1|\hat{f}_k| + \ln\lambda\sum\nolimits_5 ||q_1||_1|\hat{f}_k\lambda_k^{-1}||\lambda - |\lambda_k||^{-1} \leqslant O(\frac{1}{\lambda}). \quad (38)$$

II) $||p_1||_1 \neq 0$. Для суммы $\sum_1$ имеем оценку $O(\lambda^{-1})$. Сумму $\sum_2$ оценим следующим образом: $\lambda^{-1}\sum_2(|\lambda_k|\,||p_1||_1 + ||q_1||_1)|\hat{f}_k\lambda_k^{-1}|\ln|\lambda_k| = \varphi(\lambda)$, где $\varphi(\lambda) = O(\lambda^{-1}\ln^{-\beta+1}\lambda)$ при $\nu > 1$, $\varphi(\lambda) = O(\lambda^{-1}\ln^{-\beta+2}\lambda)$ при $\nu = 1$ и $\varphi(\lambda) = O(\lambda^{-\nu}\ln^{-\beta+1}\lambda)$ при $\nu < 1$. Суммы $\sum_3$ и $\sum_4$ имеют оценку $O(\lambda^{-\nu}\ln^{-\beta+1}\lambda)$. Для суммы $\sum_5$ получаем оценку $O(\lambda^{-\nu}\ln^{-\beta+2}\lambda)$.

Таким образом, для второй квадратной скобки в случае $||p_1||_1 \neq 0$ получаем оценку

$$\left[\lambda^{-1}\sum\nolimits_1 + \lambda^{-1}\sum\nolimits_2 + \ln\lambda\sum\nolimits_3 + \lambda^{-1}\ln\lambda\sum\nolimits_4 + \ln\lambda\sum\nolimits_5\right] \leqslant O(\max(\frac{1}{\lambda}, \frac{\ln^2\lambda}{\lambda^\nu\ln^\beta\lambda})). \quad (39)$$

Объединяя попарно оценки (37), (38) и (37), (39), получим оценку суммы $S_2(x)$ при $||p_1||_1 = 0$ и при $||p_1||_1 \neq 0$:

$$S_2(x) = O(\max(\frac{1}{\lambda},\ \frac{1}{\lambda^\nu \ln^\beta \lambda},\ ||p_1||_1 \frac{\ln^2 \lambda}{\lambda^\nu \ln^\beta \lambda})). \tag{40}$$

Оценивая сумму $S_3(x)$, содержащую коэффициент $p_1(x)$, будем использовать оценку $||u_k'||_\infty \leqslant c(1+|\lambda_k|)||u_k||_\infty$, справедливую при условиях (2) [10], [6]. По условию $p_1(x) \in \mathcal{L}^s(G)$. Рассмотрим сначала случай $s \geqslant p$, тогда $p_1 \in \mathcal{L}^p(G)$. Применим под знаком интеграла указанную оценку производной, перейдем от нормы суммы к сумме норм и применим обобщенное неравенство Минковского к $\mathcal{L}^p-$норме следующего интеграла:

$$J(x) = \int_0^R S_0[|K_0(\lambda, \lambda_k, \tau, R)|] s_\tau(|p_1(x)|)\, d\tau, \tag{41}$$

получим $||S_3||_{p,K} \leqslant c_1 ||p_1||_p \sum\limits_k |\hat{f}_k \lambda_k| S_0[A_k^1(\lambda, 0, R_0, R)]$, $p \geqslant 1$, где также воспользовались на последнем шаге соотношением (23). Сравнивая правую часть последнего неравенства с формулой (36), получаем, что для $||S_3||_{p,K}$ в случае $s \geqslant p$ имеет место оценка (40).

Пусть $s < p$. Проведем те же преобразования, что и в случае $s \geqslant p$, но к интегралу (41) применим неравенство Юнга: $||J||_{p,K} \leqslant 2||p_1||_s\, ||S_0[|K_0|]||_{\varkappa,(0,R_0)} \leqslant 2||p_1||_s S_0[A_k^\varkappa(\lambda, 0, R_0, R)]$, где также использовали соотношение (23). Параметр $\varkappa$ связан с параметрами задачи следующим образом:

$$\frac{1}{\varkappa} = 1 + \frac{1}{p} - \frac{1}{s} < 1, \quad (s < p).$$

Учитывая оценки из леммы об оценках интегралов [9] для $A_k^\varkappa$, для S_3 получаем:

$$||S_3||_{p,K} \leqslant c||p_1||_s \sum_k |\hat{f}_k \lambda_k| S_0[A_k^\varkappa(\lambda, 0, R_0, R)] \leqslant c_0 ||p_1||_s \Big[\lambda^{-1} \sum\nolimits_1 |\hat{f}_k| + \lambda^{-1} \sum\nolimits_2 |\hat{f}_k \lambda_k^{1-\frac{1}{\varkappa}}| +$$

$$+ \lambda^{1-\frac{1}{\varkappa}} \sum\nolimits_3 |\hat{f}_k \lambda_k^{-1}| + \lambda^{-\frac{1}{\varkappa}} \sum\nolimits_4 |\hat{f}_k \lambda_k| + \lambda^{1-\frac{1}{\varkappa}} \sum\nolimits_5 |\hat{f}_k| |\lambda - |\lambda_k||^{-1}\Big] \tag{42}$$

Оценим правую часть последнего неравенства по прежней схеме: $\lambda^{-1} \sum_1 |\hat{f}_k| = O(\lambda^{-1})$; $\lambda^{-1} \sum_2 |\hat{f}_k \lambda_k^{1-1/\varkappa}| = \varphi(\lambda)$, где $\varphi(\lambda) = O(\lambda^{-1})$ при $\nu > 2 - 1/\varkappa$, $\varphi(\lambda) = O(\lambda^{-1} \ln^{-\beta+1} \lambda)$ при $\nu = 2 - 1/\varkappa$ и $\varphi(\lambda) = O(\lambda^{-\nu-1/\varkappa+1}) = O(\lambda^{-\nu-1/p+1/s})$ при $\nu < 2 - 1/\varkappa$; $\lambda^{1-1/\varkappa} \sum_3 |\hat{f}_k \lambda_k^{-1}| = O(\lambda^{-\nu-1/\varkappa+1} \ln^{-\beta} \lambda) = O(\lambda^{-\nu-1/p+1/s} \ln^{-\beta} \lambda)$; такая же оценка имеет место для суммы $\sum_4$; $\lambda^{1-1/\varkappa} \sum_5 |\hat{f}_k| |\lambda - |\lambda_k||^{-1} = O(\lambda^{-\nu-1/\varkappa+1} \ln^{-\beta+1} \lambda) = O(\lambda^{-\nu-1/p+1/s} \ln^{-\beta+1} \lambda)$. Таким образом,

$$||S_3||_{p,K} = O(\max(\frac{1}{\lambda},\ \frac{\ln \lambda}{\lambda^{\nu+\frac{1}{p}-\frac{1}{s}} \ln^\beta \lambda},\ \frac{1}{\lambda^{\nu+\frac{1}{p}-\frac{1}{s}}})). \tag{43}$$

Объединяя оценки (21), (25), (40), (43), получаем оценку слагаемого Δ_λ.

Оценим слагаемое Δ_λ^a из (6) (Δ_λ^b оценивается аналогично). Положим $t = (b-a)/4$, $x = z - t$, $x \in [0, a]$. Тогда $z \in K = [t, a+t] \subset G$. В интеграле Δ_λ^a перейдем к переменной z :

$$\Delta_\lambda^a = \int_K |\sigma_\lambda(z-t, f) - S_\lambda(z-t, f)|^p\, dz = \int_K \left| \int_G f(y)[\Theta_\lambda(z-t, y) - D_\lambda(z-t, y)] dy \right|^p dz,$$

где $D_\lambda-$ ядро Дирихле, - спектральная функция оператора L_0. Положим $R_0 = t/2$ и фиксируем произвольно число $R \in [R_0/2, R_0]$. Введем вспомогательные функции

$$v_{1\lambda}(\rho_1, R) = \begin{cases} \frac{\sin \lambda \rho_1}{\pi \rho_1}, & \rho_1 \in [0, R],\ \rho_1 = z + t - y,\ y \in \overline{G}, \\ 0, & \rho_1 < 0, \rho_1 > R, \end{cases}$$

$$v_{2\lambda}(\rho_2,R)=\begin{cases}\frac{\sin\lambda\rho_2}{\pi\rho_2}, & \rho_2\in[-R,0],\ \rho_2=z-t-y,\\ 0, & \rho_2<-R,\rho_2>0,\end{cases}$$

$$v_{3\lambda}(\rho_1,R)=\begin{cases}\frac{\sin\lambda\rho_1}{\pi\rho_1}, & |\rho_1|\leqslant R,\\ 0, & |\rho_1|>R.\end{cases}$$

Применим к ним операцию усреднения по R:

$$\forall v_\lambda(\rho,R):\quad S_0[v_\lambda(\rho,R)]=\frac{8}{3R_0^2}\int\limits_{R_0/2}^{R_0}R\,v_\lambda(\rho,R)\,dR;\quad S_0[1]=1.$$

Под знаком интеграла в Δ_λ^a добавим и вычтем функцию $\widetilde{v}_0(z,y)=S_0[v_{3\lambda}(\rho_1,R)-2v_{1\lambda}(\rho_1,R)-2v_{2\lambda}(\rho_2,R)]$, используем оценку $|a+b|^p\leqslant 2^{p-1}(|a|^p+|b|^p)$; получим

$$\Delta_\lambda^a\leqslant 2^{p-1}(I+\widetilde{I}),\tag{44}$$

где

$$I=\left\|\int\limits_{\overline{G}}f(y)[\Theta_\lambda(z-t,y)+\widetilde{v}_0(z,y)]\,dy\right\|_{p,K}^p,$$

а $\widetilde{I}$ получается из I заменой Θ_λ на D_λ. Оценим интеграл I. Работа с $\widetilde{I}$ аналогична работе с I, но проще, т.к. в операторе $L_0: p_1=q_1=0$.

Для функций u_k имеет место формула среднего значения, легко получаемая интегрированием по частям:

$$u_k(z+t)+u_k(z-t)=2u_k(z)\cos\lambda_k t-h_k(z,t),\quad \lambda_k\in\mathcal{C},\ z\pm t\in\overline{G},\tag{45}$$

где

$$h_k(z,t)=\frac{1}{\lambda_k}\int\limits_{z-t}^{z+t}A_1u_k(\tau)\sin\lambda_k(|z-\tau|-t)\,d\tau,\quad \left.\frac{\sin\lambda_k(|z-\tau|-t)}{\lambda_k}\right|_{\lambda_k=0}=|z-\tau|-t;$$

$A_1u_k(x)=A_1\overset{m}{u}_k(x)=\mu_0\overset{m-1}{u}_k(x)-p_1(x)\overset{m}{u}_k'(x)-q_1(x)\overset{m}{u}_k(x)$, $\overset{m}{u}_k-$ m-я присоединенная функция ($m=0$ - собственная функция, $\overset{-1}{u}\equiv 0$).

Положим $Tu_k(z,t)=2u_k(z)\cos\lambda_k t-h_k(z,t)$. Формулу (45) можно переписать так: $u_k(z-t)=-u_k(z+t)+Tu_k(z,t)$.

Тогда $\Theta_\lambda(z-t,y)=-\Theta_\lambda(z+t,y)+\sum^\lambda Tu_k(z,t)\bar{v}_k(y)$. Подставим это выражение в I и используем неравенство, примененное в (44):

$$I\leqslant 2^{p-1}\left\|\int\limits_{\overline{G}}f(y)\big[\sum^\lambda Tu_k(z,t)\bar{v}_k(y)-v_0(z,y)\big]\,dy\right\|_{p,K}^p+$$

$$+2^{p-1}\left\|\int\limits_{\overline{G}}f(y)\big[\Theta_\lambda(z+t,y)-S_0[v_{3\lambda}(\rho_1,R)]\big]\,dy\right\|_{p,K}^p\equiv 2^{p-1}(I_1^p+I_2^p),\tag{46}$$

где $v_0(z,y)=2S_0[v_{1\lambda}(\rho_1,R)+v_{2\lambda}(\rho_2,R)]$. Рассмотрим интеграл I_1.

В работе И.С. Ломова [7] доказана следующая

Лемма 2 *Фиксируем $p \in [1,\infty)$. Пусть выполняются условия I и условие (1), $\lambda > 2/R_0-$ достаточно большое число. Тогда $\forall f(x) \in \mathcal{L}^r(G)$: $\lim\limits_{\Lambda\to\infty} \Delta(\Lambda) = 0$, где*

$$\Delta(\Lambda) = \Big|\Big|\int\limits_{\overline{G}} f(y)\big[v_0(z,y) - \sum\nolimits^{\Lambda}(v_0,\bar{u}_k)(z)\bar{v}_k(y)\big]\,dy\Big|\Big|_{p,K}.$$

Получено представление для коэффициентов $(v_0,\bar{u}_k)$:

$$(v_0,\bar{u}_k)(z) = 2u_k(z)\delta_k^{\lambda}\cos\lambda_k t + 2u_k(z)I_k^{\lambda}(R_0) + 2u_k(z)S_0[I_R(\lambda,\lambda_k)]\sin\lambda_k t -$$

$$-\frac{2}{\pi}S_0\Big[\int\limits_0^R \frac{\sin\lambda r}{r}h_k(z,t-r)\,dr\Big], \tag{47}$$

где $\delta_k^{\lambda} = 0$ при $|\lambda_k| > \lambda$, $I_k^{\lambda}(R_0) = O(\lambda_k^{-2})$ при $|\lambda_k| \geqslant 3\lambda/2$, $I_R = O(\lambda\lambda_k^{-1}) = O(\lambda_k^{-1})$ при $|\lambda_k| \geqslant 3\lambda/2$; $\quad S(z) = \frac{2}{\pi}S_0\big[\int\limits_0^R \frac{\sin\lambda r}{r}h_k(z,t-r)\,dr\big], I_R = \frac{2}{\pi}\int\limits_0^R \frac{\sin\lambda r\sin\lambda_k r}{r}dr.$

Оценим интеграл I_1 из (46). Перейдем от $\mathcal{L}^p$ к $\mathcal{L}^q-$норме $(p^{-1}+q^{-1}=1)$:

$$I_1 \leqslant \Big|\Big|\int\limits_{\overline{G}} f(y)\big[\sum\nolimits^{\lambda} Tu_k(z,t)\bar{v}_k(y) - v_0(z,y)\big]\,dy\Big|\Big|_{q,K},$$

заменим функцию v_0 биортогональным рядом, используем структуру коэффициентов (47) и оценим модуль суммы через сумму модулей, получаем

$$I_1 \leqslant \Big|\Big|2\sum_{k=1}^{\infty}\big|f_k u_k(z)I_k^{\lambda}(R_0)\big| + 2\Big|\sum_{k=1}^{\infty} f_k u_k(z)S_0[I_R(\lambda,\lambda_k)]\sin\lambda_k t\Big| +$$

$$+\Big|\sum_{k=1}^{\infty} f_k S(z) - \sum\nolimits^{\lambda} f_k h_k(z,t)\Big|\Big|\Big|_{q,K}. \tag{48}$$

Первое слагаемое в последнем неравенстве совпадает с (20), поэтому получаем для него справедлива оценка (21). Второе слагаемое в (48) оценим следующим образом:

$$\Big|\Big|\sum_{k=1}^{\infty} f_k u_k(z)S_0[I_R(\lambda,\lambda_k)]\sin\lambda_k t\Big|\Big|_q \leqslant c\Big|\Big|\hat{f}_k S_0[I_R]\Big|\Big|_q \leqslant c\Bigg[\lambda^{-q}\sum\nolimits_1 |\hat{f}_k|^q +$$

$$+\lambda^{-q}\sum\nolimits_2 |\hat{f}_k\lambda_k|^q + \lambda^q\sum\nolimits_3 |\hat{f}_k\lambda_k^{-1}|^q + \ln^q\lambda\sum\nolimits_4 |\hat{f}_k{}^q| + \sum\nolimits_5 |\hat{f}_k|^q\ln^q\frac{\lambda}{|\lambda-|\lambda_k||}\Bigg]^{1/q}. \tag{49}$$

Для суммы $\sum_1$ получим оценку $O(\lambda^{-q})$. Сумму $\sum_2$ оценим следующим образом:
$\lambda^{-q}\sum_2 |\hat{f}_k\lambda_k|^q \leqslant c\lambda^{-q}\sum\limits_{n=2}^{[\lambda/2]} n^{(-\nu+1)q}\ln^{-\beta q} n = \varphi(\lambda)$, где $\varphi(\lambda) = O(\lambda^{-q})$ при $\nu > 1+1/q$, $\varphi(\lambda) = O(\lambda^{-q}\ln^{-\beta q+1}\lambda)$ при $\nu = 1+1/q$ и $\varphi(\lambda) = O(\lambda^{-\nu q+1})$ при $\nu < 1+1/q$. Для суммы $\sum_3$ получаем $\lambda^q\sum_3 |\hat{f}_k\lambda_k^{-1}|^q \leqslant O(\lambda^{-\nu q+1}\ln^{-\beta q}\lambda)$. Сумма $\sum_4$ имеет, очевидно, оценку $O(\lambda^{-\nu q}\ln^{-\beta q+q}\lambda)$, а сумма $\sum_5 |\hat{f}_k|^q\ln^q\frac{\lambda}{|\lambda-|\lambda_k||} \leqslant O(\lambda^{-\nu q+1})$.

Объединяя полученные оценки и вычисляя корень степени q, получаем оценку выражения (49):

$$O(\max(\frac{1}{\lambda}, \frac{1}{\lambda^{\nu+\frac{1}{p}-1}})). \tag{50}$$

В работах И.С. Ломова [7] показано, что для оценки третьего слагаемого в (48) справедлива полученная ранее оценка для $S(x)$ - формула (22), которая в свою очередь складывается из оценок (25),(40) и (43).

Объединяя полученные оценки всех трех слагаемых (48), получаем оценку интеграла I_1. Для интеграла I_2 из (46) проводим те же преобразования, что и для I_1. Ситуация здесь даже проще, так как не появляется интеграл $I_R(\lambda, \lambda_k)$. Для интеграла I (а значит и $\widetilde{I}$) из (44) получаем ту же оценку, что и для I_1. Эта оценка, тем самым, верна и для Δ_λ^a из (6). Преобразования для Δ_λ^b ничем не отличаются от Δ_λ^a. Интеграл Δ_λ из (6) оценен с лучшими показателями в соответствующих суммах, чем Δ_λ^a. Таким образом, полученная для I_1 оценка имеет место для всей разности (6) и является искомой оценкой (5) Теоремы 1.

4 Характер зависимости оценок скорости равносходимости от расстояния от компакта до границы основного интервала

Теорема 2. *Пусть для оператора L и функции $f(x)$ выполняются условия (1),(4′) и условия I. Тогда для всех достаточно больших чисел λ и любого отрезка $K \subset G$ справедлива оценка*

$$\Big|\Big|\sigma_\lambda(x,f) - S_\lambda(x,f)\Big|\Big|_{\mathcal{L}^p(K)} \leqslant \frac{1}{R_0}\, O\left(\max\left(\frac{1}{\lambda},\, \frac{\ln\lambda}{\lambda^\nu},\, ||p_1||_1\Big[\frac{\ln\lambda}{\lambda^{\nu+\frac{1}{p}-\frac{1}{s}}},\, \frac{n_1 \ln^2\lambda}{\lambda^\nu}\Big]\right)\right), \qquad (51)$$

где $c = const$ не зависит от λ, R_0; $\quad R_0 \in (0, \frac{1}{2}\, dist(K, \partial G))$; $\quad n_1 = 1$, если присутствуют присоединенные функции и $||p_1||_1 \neq 0$, и $n_1 = 0$ в противном случае.

Доказательство.

Доказательство в целом повторяет получение оценки слагаемого Δ_λ из Теоремы 1. Остановимся подробнее на лемме о разрывном множителе Дирихле в $\mathcal{C}$ (оценки (17)).

$$\frac{2}{\pi} S_0\left[\int_0^R \frac{\sin\lambda t \cos\lambda_k t}{t} dt\right] = \delta_k^\lambda + I_k^\lambda(R_0), \quad I_k^\lambda(R_0) = \frac{2}{\pi}\, S_0[\widetilde{I}_k^\lambda], \text{ где } \widetilde{I}_k^\lambda = \int_R^\infty \frac{\sin\lambda t \cos\lambda_k t}{t} dt. \quad (52)$$

Преобразуем в интеграле $\widetilde{I}_k^\lambda$ произведение тригонометрических функций в сумму синусов, затем проинтегрируем по частям:

$$\widetilde{I}_k^\lambda = \frac{\cos(\lambda-\lambda_k)R}{2(\lambda-\lambda_k)R} - \frac{1}{2(\lambda-\lambda_k)} \int_R^\infty \frac{\cos(\lambda-\lambda_k)t}{t^2}\, dt +$$

$$+ \frac{\cos(\lambda+\lambda_k)R}{2(\lambda+\lambda_k)R} - \frac{1}{2(\lambda+\lambda_k)} \int_R^\infty \frac{\cos(\lambda+\lambda_k)t}{t^2}\, dt.$$

Усредним первое слагаемое в выражении для $\widetilde{I}_k^\lambda$:

$$\frac{8}{3R_0^2} \int_{R_0/2}^{R_0} \frac{\cos(\lambda-\lambda_k)R}{2(\lambda-\lambda_k)R}\, R\, dR = \frac{4}{3R_0^2\,(\lambda-\lambda_k)^2}\, \sin(\lambda-\lambda_k)R\Big|_{R_0/2}^{R_0} =$$

$$= \frac{8}{3R_0^2(\lambda-\lambda_k)^2}\, \sin\frac{(\lambda-\lambda_k)R_0}{4}\, \cos\frac{3(\lambda-\lambda_k)R_0}{4}.$$

Воспользовавшись оценками $|\sin z| \leqslant |z|$ и $|\cos z| \leqslant c$, получаем, что первое слагаемое интеграла I_k^λ имеет оценку $O\big((R_0\,|\lambda - |\lambda_k||)^{-1}\big)$.

Второе слагаемое $\widetilde{I}_k^\lambda$ оценим сверху

$$\left|\frac{1}{2(\lambda-\lambda_k)}\int\limits_R^\infty \frac{\cos{(\lambda-\lambda_k)t}}{t^2}\,dt\right| \leqslant \left|\frac{1}{2(\lambda-\lambda_k)}\,\frac{1}{R}\right|$$

и усредним по R:

$$\frac{8}{3R_0^2}\,\frac{1}{2(\lambda-\lambda_k)}\int\limits_{R_0/2}^{R_0}\frac{1}{R}\,R\,dR = \frac{8}{3R_0^2}\,\frac{1}{2(\lambda-\lambda_k)}\,\frac{R_0}{2} = O\left(\frac{1}{R_0(\lambda-\lambda_k)}\right).$$

Третье и четвертое слагаемые в выражении для $\widetilde{I}_k^\lambda$ усредняются аналогично, а с учетом того, что $|\lambda+\lambda_k| \geqslant |\lambda - |\lambda_k||$, $\big|\lambda+\lambda_k\big|^{-1} \leqslant \big|\lambda-|\lambda_k|\big|^{-1}$, получаем оценку

$$I_k^\lambda = O\left(\frac{1}{R_0|\lambda-|\lambda_k||}\right) \quad \text{при} \quad R_0 \to 0, \quad |\lambda-|\lambda_k|| \leqslant const.$$

Далее получим оценки интеграла $I_k^\lambda(R_0)$, аналогичные оценкам, полученным И.С. Ломовым в лемме об оценках интегралов [9]:

1) при $|\lambda_k| \leqslant 1$ $\qquad I_k^\lambda = O\left(\dfrac{1}{R_0\lambda}\right)$,

2) при $1 \leqslant |\lambda_k| \leqslant \lambda/2$ $\qquad I_k^\lambda = O\left(\dfrac{1}{R_0\lambda}\right)$,

3) при $|\lambda_k| \geqslant 3\lambda/2$ $\qquad I_k^\lambda = O\left(\dfrac{1}{R_0|\lambda_k|}\right)$,

4) при $|\lambda-|\lambda_k|| \leqslant 2/R_0$ $\qquad I_k^\lambda = O\,(1)$,

5) при $2/R_0 \leqslant |\lambda-|\lambda_k|| \leqslant \lambda/2$ $\qquad I_k^\lambda = O\left(\dfrac{1}{R_0|\lambda-|\lambda_k||}\right)$.

В соответствии с полученными оценками, формула (20) примет вид

$$\sum_k \Big|f_k u_k(x) I_k^\lambda(R_0)\Big| \leqslant c\Big[\frac{1}{R_0\lambda}\sum\nolimits_1\big|\hat{f}_k\big| + \frac{1}{R_0\lambda}\sum\nolimits_2\big|\hat{f}_k\big| + \frac{1}{R_0}\sum\nolimits_3\big|\hat{f}_k\lambda_k^{-1}\big| + \sum\nolimits_4\big|\hat{f}_k\big| + \\ + \frac{1}{R_0}\sum\nolimits_5\frac{\big|\hat{f}_k\big|}{\big|\lambda-|\lambda_k|\big|}\Big]. \tag{53}$$

Оценим суммы в правой части последнего неравенства. Для суммы $\dfrac{1}{R_0\lambda}\sum_1\big|\hat{f}_k\big|$ имеем оценку $O\big(\dfrac{1}{R_0\lambda}\big)$. Вторую сумму оценим следующим образом: $\dfrac{1}{R_0\lambda}\sum_2\big|\hat{f}_k\big| \leqslant \dfrac{c}{R_0\lambda}\sum_2|\lambda_k|^{-\nu} \leqslant$ $\leqslant \dfrac{c}{R_0\lambda}\sum\limits_{n=1}^{[\lambda/2]}\dfrac{1}{n^\nu}\sum\limits_{0\leqslant|\lambda_k-n\leqslant 1} 1 = \varphi(\lambda,R_0)$, где $\varphi(\lambda,R_0) = O(\dfrac{1}{R_0\lambda^\nu})$ при $\nu<1$, $\quad\varphi(\lambda,R_0) = O(\dfrac{\ln\lambda}{R_0\lambda})$ при $\nu=1$ и $\varphi(\lambda,R_0) = O(\dfrac{1}{R_0\lambda})$ при $\nu>1$. Для третьей суммы справедлива оценка

$\frac{1}{R_0}\sum_3\left|\hat{f}_k\lambda_k^{-1}\right| \leqslant \frac{c}{R_0}\sum\limits_{n\geqslant[3\lambda/2]}\frac{1}{n^{\nu+1}} = O(\frac{1}{R_0\lambda^\nu})$. Сумма $\sum_4\left|\hat{f}_k\right|$ имеет, очевидно, оценку $O(\frac{1}{\lambda^\nu})$, а сумма $\sum_5$:

$$\frac{1}{R_0}\sum\nolimits_5\frac{\left|\hat{f}_k\right|}{\left|\lambda-|\lambda_k|\right|} \leqslant \frac{c}{R_0}\sum\nolimits_5\frac{1}{|\lambda_k|^\nu|\lambda-|\lambda_k||} = O(\frac{\ln\lambda}{R_0\lambda^\nu}).$$

Объединяя полученные оценки, получаем

$$\sum_k\left|f_ku_k(x)I_k^\lambda(R_0)\right| = O\left(\max\left(\frac{\ln\lambda}{R_0\lambda^\nu},\ \frac{1}{R_0\lambda}\right)\right). \tag{54}$$

Для суммы $S_2(x)$ запишем выражение (36):

$$|S_2(x)| \leqslant c\sum_k|\hat{f}_k|\left[I_{\lambda1}+(|\lambda_k|\,||p_1||_1+||q_1||_1)S_0[A_k^1(\lambda,0,R_0,R)]\right] \leqslant$$

$$\leqslant \frac{c_0}{R_0}\left[\lambda^{-1}\sum\nolimits_1|\hat{f}_k|+\lambda^{-2}\sum\nolimits_2|\hat{f}_k|+\lambda^{-1}\sum\nolimits_3|\hat{f}_k\lambda_k^{-1}|+\sum\nolimits_4|\hat{f}_k|+\sum\nolimits_5|\hat{f}_k||\lambda-|\lambda_k||^{-2}\right]+$$

$$+c_0\left[\lambda^{-1}\sum\nolimits_1|\hat{f}_k|+\lambda^{-1}\sum\nolimits_2(|\lambda_k|\,||p_1||_1+||q_1||_1)|\hat{f}_k\lambda_k^{-1}|\ln|\lambda_k|+\ln\lambda\sum\nolimits_3(|\lambda_k|\,||p_1||_1+||q_1||_1)|\hat{f}_k\lambda_k^{-2}|+\right.$$

$$\left.+\lambda^{-1}\ln\lambda\sum\nolimits_4(|\lambda_k|\,||p_1||_1+||q_1||_1)|\hat{f}_k|+\ln\lambda\sum\nolimits_5(|\lambda_k|\,||p_1||_1+||q_1||_1)|\hat{f}_k\lambda_k^{-1}||\lambda-|\lambda_k||^{-1}\right].$$

Для первой квадратной скобки в правой части последнего неравенства при условиях (2), (4') получаем оценку $O(\max(\frac{1}{R_0\lambda},\ \frac{1}{R_0\lambda^\nu}))$. Для второй скобки - $O(\max(\frac{\ln^2\lambda}{\lambda^\nu},\ \frac{1}{\lambda}))$, $\quad||p_1||_1\neq0$.

Объединяя оценки, получаем оценку для всей суммы S_2:

$$S_2(x) = O(\max(\frac{1}{R_0\lambda},\ \frac{\ln^2\lambda}{R_0\lambda^\nu})), \quad ||p_1||_1\neq0; \tag{55}$$

$$S_2(x) = O(\max(\frac{1}{R_0\lambda},\ \frac{1}{R_0\lambda^\nu})), \quad ||p_1||_1=0. \tag{56}$$

Оценки $||S_1||_{p,K}$ и $||S_3||_{p,K}$ равномерны по $x\in K$, поэтому не зависят от R_0.

Объединяя оценки (54), (55), (56), (24) и (42) получаем оценки Теоремы 2.

Замечание. Оценки (24) и (42) в случае, когда нормированные коэффициенты Фурье f_k имеют асимптотику (4′), получены в работе И.С. Ломова [9].

В случае замены условия (4′) Теоремы 2 на условие (4), имеет место аналогичная

Теорема 3. *Пусть для оператора L и функции $f(x)$ выполняются условия (1),(4) и условия I. Тогда для всех достаточно больших чисел λ и любого отрезка $K\subset G$ справедлива оценка*

$$\left|\left|\sigma_\lambda(x,f)-S_\lambda(x,f)\right|\right|_{\mathcal{L}^p(K)} \leqslant$$

$$\leqslant \frac{1}{R_0}O\left(\max\left(\frac{1}{\lambda},\ \frac{1}{\lambda^\nu},\ \frac{\ln\lambda}{\lambda^\nu\ln^\beta\lambda},\ ||p_1||_1\left[\frac{\ln\lambda}{\lambda^{\nu+\frac{1}{p}-\frac{1}{s}}\ln^\beta\lambda},\ \frac{1}{\lambda^{\nu+\frac{1}{p}-\frac{1}{s}}},\ \frac{n_1\ln^2\lambda}{\lambda^\nu\ln^\beta\lambda}\right]\right)\right), \tag{57}$$

где $c=const$ не зависит от λ, R_0; $\quad R_0\in(0,\frac{1}{2}\,dist(K,\partial G))$; $\quad n_1=1$, если присутствуют присоединенные функции и $||p_1||_1\neq0$, и $n_1=0$ в противном случае.

Доказательство.

При доказательстве этой теоремы будут использоваться оценки Теоремы 1 (суммы, отвечающие потенциалу $q_1(x)$ и коэффициенту при первой производной $p_1(x)$ - S_1 и S_3 соответственно). Повторяются рассуждения, приведенные в Теореме 2.

Для суммы, отвечающей интегралу $I_k^\lambda(R_0)$, запишем формулу (53):

$$\sum_k \left|f_k u_k(x) I_k^\lambda(R_0)\right| \leqslant c\Bigg[\frac{1}{R_0\lambda}\sum\nolimits_1 |\hat{f}_k| + \frac{1}{R_0\lambda}\sum\nolimits_2 |\hat{f}_k| + \frac{1}{R_0}\sum\nolimits_3 |\hat{f}_k\lambda_k^{-1}| + \sum\nolimits_4 |\hat{f}_k| + \\ + \frac{1}{R_0}\sum\nolimits_5 \frac{|\hat{f}_k|}{|\lambda - |\lambda_k||}\Bigg].$$

Оценим слагаемые, стоящие в правой части неравенства, используя условия теоремы: Для суммы $\frac{1}{R_0\lambda}\sum_1|\hat{f}_k|$ имеем оценку $O(\frac{1}{R_0\lambda})$. Вторую сумму оценим следующим образом: $\frac{1}{R_0\lambda}\sum_2|\hat{f}_k| \leqslant \frac{c}{R_0\lambda}\sum_2 |\lambda_k|^{-\nu}\ln^{-\beta}|\lambda_k| = \varphi(\lambda, R_0)$, где $\varphi(\lambda, R_0) = O(\frac{1}{R_0\lambda^\nu})$ при $\nu < 1$, $\varphi(\lambda, R_0) = O(\frac{\ln\lambda}{R_0\lambda\ln^\beta\lambda})$ при $\nu = 1$ и $\varphi(\lambda, R_0) = O(\frac{1}{R_0\lambda})$ при $\nu > 1$. Для третьей суммы справедлива оценка $\frac{1}{R_0}\sum_3|\hat{f}_k\lambda_k^{-1}| \leqslant \frac{c}{R_0}\sum\limits_{n\geqslant[3\lambda/2]}\frac{1}{n^{\nu+1}\ln^\beta n} = O(\frac{1}{R_0\lambda^\nu\ln^\beta\lambda})$. Сумма $\sum_4|\hat{f}_k|$ имеет, очевидно, оценку $O(\frac{1}{\lambda^\nu\ln^\beta\lambda})$, а сумма $\sum_5$:

$$\frac{1}{R_0}\sum\nolimits_5 \frac{|\hat{f}_k|}{|\lambda - |\lambda_k||} = O(\frac{\ln\lambda}{R_0\lambda^\nu\ln^\beta\lambda}).$$

Объединяя полученные оценки, получаем

$$\sum_k \left|f_k u_k(x) I_k^\lambda(R_0)\right| = O\left(\max\left(\frac{\ln\lambda}{R_0\lambda^\nu\ln^\beta\lambda}, \frac{1}{R_0\lambda}, \frac{1}{R_0\lambda^\nu}\right)\right). \tag{58}$$

Для суммы $S_2(x)$ выполняется оценка (40) с точностью до множителя $\frac{1}{R_0}$:

$$S_2(x) = \frac{1}{R_0}O(\max(\frac{1}{\lambda}, \frac{1}{\lambda^\nu\ln^\beta\lambda}, ||p_1||_1\frac{\ln^2\lambda}{\lambda^\nu\ln^\beta\lambda})). \tag{59}$$

Объединяя оценки (58), (59), (25) и (43), получаем оценки Теоремы 3.

Автор признателен своему научному руководителю И.С. Ломову за постановку задачи, ценные замечания и большое внимание к данной работе.

Список литературы

[1] *Бесов О.В., Ильин В.П., Никольский С.М.* Интегральные представления функций и теоремы вложения. М.: Наука. 1975.

[2] *Зигмунд А.* Тригонометрические ряды. М.: Мир. 1965. Т. 1,2.

[3] *Ильин В.А.* Необходимые и достаточные условия базисности и равносходимости с тригонометрическим рядом спектральных разложений. I, II// Дифференц. уравнения. 1980. Т. 16, № 5. С. 771-794. Т. 16, № 6. С. 980-1009.

[4] *Ильин В.А.* Необходимые и достаточные условия базисности в $\mathcal{L}^p$ и равносходимости с тригонометрическим рядом спектральных разложений и разложений по системам экспонент// ДАН СССР. 1983. Т. 273, № 4. С. 789-793.

[5] *Ильин В.А.* Равносходимость с тригонометрическим рядом разложений по корневым функциям одномерного оператора Шредингера с комплексным потенциалом из класса $\mathcal{L}^1$// Дифференц. уравнения. 1991. Т. 27, № 4. С. 577-597.

[6] *Ломов И.С.* Некоторые свойства спектральных разложений, связанных с операторами типа Штурма-Лиувилля// ДАН СССР. 1979. Т. 248, № 5. С. 1063-1065.

[7] *Ломов И.С.* О скорости сходимости биортогональных рядов, связанных с дифференциальными операторами второго порядка// Дифференц. уравнения. 1996. Т. 32, № 1. С. 58-69.

[8] *Ломов И.С.* О скорости сходимости биортогональных разложений функций// Дифференц. уравнения. 1996. Т. 32, № 12. С. 1618-1629.

[9] *Ломов И.С.* О влиянии степени суммируемости коэффициентов дифференциальных операторов на скорость равносходимости спектральных разложений. I, II// Дифференц. уравнения. 1998. Т. 34, № 5. С. 619-628. Т. 34, № 8. С. 1066-1077.

[10] *Ломов И.С.* Свойство базисности корневых векторов нагруженных дифференциальных операторов второго порядка//Дифференц. уравнения. 1991. Т. 27, № 1. С. 80-93.

УДК 519.254

РАЗДЕЛЕНИЕ СМЕСЕЙ ВЕРОЯТНОСТНЫХ РАСПРЕДЕЛЕНИЙ СЕТОЧНЫМ МЕТОДОМ МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ ПРИ ПОМОЩИ АЛГОРИТМА УСЛОВНОГО ГРАДИЕНТА

© 2009 г. А. Л. Назаров

nazarov.vmik@gmail.com

Кафедра Математической статистики

1 Введение.

Конечные смеси вероятностных законов находят самое широкое применение как модели распределений многих величин, наблюдаемых на практике. Например, смеси нормальных законов используются в финансовой математике, при статистической обработке турбулентной плазмы(см.[2]). В этих областях требуется высокая скорость работы соответствующих методов.

Для решения задачи разделения смесей традиционно используется EM-алгоритм. Однако он не обладает достаточным быстродействием и устойчивостью по входным данным(см., например, [3]).

В данной статье рассматривается метод приближенного решения задачи разделения смесей с помощью сеточного метода максимального правдоподобия, заключающегося в минимизации функции правдоподобия смеси, число компонент которой заведомо больше, чем в исходной. В качестве алгоритма минимизации функции правдоподобия использован метод условного градиента.

2 Основная идея сеточных методов разделения смесей вероятностных распределений

Идея сеточных методов подробно изложена в работе[2]. Она заключается в следующем.

Всюду далее стандартная нормальная функция распределения будет обозначаться $\Phi(x)$. Рассмотрим смесь функций распределения вида

$$F(x) = \sum_{i=1}^{k} p_i \Phi\Big(\frac{x - a_i}{\sigma_i}\Big), \qquad x \in \mathbb{R}, \tag{1}$$

где $k \geq 1$ – целое. В классической задаче разделения смесей параметрами, подлежащими статистическому оцениванию, являются тройки (p_i, a_i, σ_i), $i = 1, \ldots, k$, где $a_i \in \mathbb{R}$, $\sigma_i > 0$, $p_i \geq 0$, $p_1 + \ldots + p_k = 1$.

Предположим, что заранее известны числа $\underline{a}$, $\overline{a}$ и $\overline{\sigma}$ такие, что $\underline{a} \leq a_i \leq \overline{a}$ и $\sigma_i \leq \overline{\sigma}$ при всех $i = 1, \ldots, k$. Другими словами, известны диапазоны изменения неизвестных параметров a_i и σ_i.

Идея, лежащая в основе рассматриваемого подхода, заключается в замене *интервалов* $[\underline{a}, \overline{a}]$ и $(0, \overline{\sigma}]$ возможных значений неизвестных параметров масштаба σ_i и сдвига a_i *дискретными* множествами известных точек. Эти точки могут быть выбраны, например, исходя из следующих соображений.

Пусть ε_a и ε_σ – положительные числа, определяющие априорные требования к точности оценивания параметров a_i и σ_i:

$$\max_i |a_i - \widehat{a}_i| \leq \varepsilon_a, \qquad \max_i |\sigma_i - \widehat{\sigma}_i| \leq \varepsilon_\sigma, \tag{2}$$

где $\widehat{a}_i$ и $\widehat{\sigma}_i$ – искомые оценки параметров. Числа ε_a и ε_σ также можно интерпретировать как пороги различимости возможных значений параметров: значения a', a'' и σ', σ'', соответственно, считаются неразличимыми, если

$$|a' - a''| \le \varepsilon_a, \qquad |\sigma' - \sigma''| \le \varepsilon_\sigma. \tag{3}$$

Положим $k_a = [(\overline{a} - \underline{a})/\varepsilon_a] + 1$, $k_\sigma = [\overline{\sigma}/\varepsilon_\sigma] + 1$, где символ $[z]$ обозначает целую часть числа z. Для $r = 1, 2, \ldots, k_a + 1$ положим $\widetilde{a}_r = \underline{a} + (r-1)\varepsilon_a$. Аналогично, для $l = 1, 2, \ldots, k_\sigma$ положим $\widetilde{\sigma}_l = l\varepsilon_\sigma$. Тогда точки с координатами $(\widetilde{a}_r, \widetilde{\sigma}_l)$ образуют узлы конечной сети, накрывающей прямоугольник $\{(a, \sigma) : \underline{a} \le a \le \overline{a},\ 0 \le \sigma \le \overline{\sigma}\}$, представляющий собой множество возможных значений параметров сдвига и масштаба компонент смеси (1) (чтобы избежать возможной некорректности, мы исключили возможность равенства параметра масштаба нулю). Число узлов полученной сети равно $K = (k_a + 1)k_\sigma$. Для удобства записи и упрощения обозначений перенумеруем каким-либо образом узлы указанной сети, вводя *единый* индекс i для координат $(\widetilde{a}_i, \widetilde{\sigma}_i)$ узла с номером i после перенумерации, $i = 1, \ldots, K$.

Базовая посылка нашего подхода заключается в аппроксимации смеси (1) смесью с заведомо бóльшим числом *известных* компонент:

$$F(x) = \sum_{i=1}^{k} p_i \Phi\Big(\frac{x - a_i}{\sigma_i}\Big) \approx \sum_{i=1}^{K} \widetilde{p}_i \Phi\Big(\frac{x - \widetilde{a}_i}{\widetilde{\sigma}_i}\Big) \equiv \widetilde{F}(x), \quad x \in \mathbb{R}. \tag{4}$$

Такое приближение практически допустимо, поскольку в силу соотношений (2) и (3) для любой пары (a_r, σ_r) параметров компоненты смеси (1) обязательно найдется практически не отличимая от нее пара $(\widetilde{a}_i, \widetilde{\sigma}_i)$ параметров компоненты смеси $\widetilde{F}(x)$. Веса же остальных компонент смеси $\widetilde{F}(x)$, для параметров которых не найдется "близкой" пары параметров (a_r, σ_r) компоненты смеси (1), можно считать равными нулю. Действительно, если бы в соотношении (4) вместо *приближенного* было бы *точное* равенство, то в силу идентифицируемости семейства конечных смесей нормальных законов, в полном соответствии с определением идентифицируемости конечных смесей с точностью до переиндексации были бы справедливы равенства:

$$k = K, \quad p_i = \widetilde{p}_i, \quad a_i = \widetilde{a}_i, \quad \sigma_i = \widetilde{\sigma}_i, \quad i = 1, \ldots, k.$$

Заметим, что неизвестными параметрами смеси $\widetilde{F}(x)$ являются *только* веса $\widetilde{p}_1, \ldots, \widetilde{p}_K$.

Пусть $\boldsymbol{x} = (x_1, \ldots, x_n)$ – (независимая) выборка наблюдений, каждое из которых представляет собой реализацию случайной величины с функцией распределения $F(x)$, задаваемой соотношением (1).

3 Разделение конечных смесей нормальных распределений с фиксированными компонентами при помощи «сеточного» метода максимального правдоподобия.

Для весов $\tilde{p}_i$, $i = 1, \ldots, K$, смеси $\tilde{F}(x)$ (см. (4)) в методе фиксированных компонент можно получить оценки максимального правдоподобия.

Пусть, как и ранее, $\boldsymbol{x} = (x_1, \ldots, x_n)$ – выборка, $(\tilde{a}_i, \tilde{\sigma}_i)$ – узлы сетки, накидываемой на множество значений параметров компонент, $i = 1, \ldots, K$. Для удобства обозначим

$$x_{ji} = \frac{x_j - \tilde{a}_i}{\tilde{\sigma}_i}, \quad \phi_{ij} = \frac{1}{\tilde{\sigma}_i}\phi(x_{ji}), \quad j = 1, \ldots, n;\ i = 1, \ldots, K,$$

где, $\phi(x)$ – стандартная нормальная плотность.

Логарифмическая «сеточная» функция правдоподобия имеет вид

$$L(\boldsymbol{p}; \boldsymbol{x}) = \log \prod_{j=1}^{n} \sum_{i=1}^{K} \tilde{p}_i \phi_{ij} = \sum_{j=1}^{n} \log\Big(\sum_{i=1}^{K} \tilde{p}_i \phi_{ij}\Big),$$

где

$$\boldsymbol{p} = (\tilde{p}_1, \ldots, \tilde{p}_K)^\top, \tag{5}$$

причем областью определения функции $L(\boldsymbol{p}; \boldsymbol{x})$ является стандартный K-симплекс

$$\mathcal{P} = \big\{\boldsymbol{p} = (\tilde{p}_1, \ldots, \tilde{p}_K)^\top\colon\ \tilde{p}_i \geq 0,\ \tilde{p}_1 + \ldots + \tilde{p}_K = 1\big\}. \tag{6}$$

Легко показать, что при любой фиксированной выборке $\boldsymbol{x}$ логарифмическая «сеточная» функция правдоподобия является вогнутой как функция весов, то есть

$$L\big(\alpha\boldsymbol{p}^{(1)} + (1-\alpha)\boldsymbol{p}^{(2)}; \boldsymbol{x}\big) \geq \alpha L(\boldsymbol{p}^{(1)}; \boldsymbol{x}) + (1-\alpha)L(\boldsymbol{p}^{(2)}; \boldsymbol{x}), \tag{7}$$

для любых точек $\boldsymbol{p}^{(1)} = (\tilde{p}_1^{(1)}, \ldots, \tilde{p}_K^{(1)})^\top$ и $\boldsymbol{p}^{(2)} = (\tilde{p}_1^{(2)}, \ldots, \tilde{p}_K^{(2)})^\top$ множества $\mathcal{P}$ и любого $\alpha \in [0,1]$.

Действительно,

$$L\big(\alpha\boldsymbol{p}^{(1)} + (1-\alpha)\boldsymbol{p}^{(2)}; \boldsymbol{x}\big) = \sum_{j=1}^{n} \log\left[\sum_{i=1}^{K}\big(\alpha\tilde{p}_i^{(1)} + (1-\alpha)\tilde{p}_i^{(2)}\big)\phi_{ij}\right] =$$

$$= \sum_{j=1}^{n} \log\left[\alpha\left(\sum_{i=1}^{K}\tilde{p}_i^{(1)}\phi_{ij}\right) + (1-\alpha)\left(\sum_{i=1}^{K}\tilde{p}_i^{(2)}\phi_{ij}\right)\right].$$

Логарифмическая функция вогнута, то есть для любых $y_1 > 0$, $y_2 > 0$ и любого $\alpha \in [0,1]$ выполнено неравенство

$$\log\big(\alpha y_1 + (1-\alpha)y_2\big) \geq \alpha \log y_1 + (1-\alpha)\log y_2. \tag{8}$$

Поэтому, продолжая приведенную выше цепочку соотношений с учетом неравенства (8) при

$$y_1 = \sum_{i=1}^{K}\tilde{p}_i^{(1)}\phi_{ij}, \qquad y_2 = \sum_{i=1}^{K}\tilde{p}_i^{(2)}\phi_{ij},$$

будем иметь

$$L\big(\alpha\boldsymbol{p}^{(1)} + (1-\alpha)\boldsymbol{p}^{(2)}; \boldsymbol{x}\big) \geq$$

$$\geq \sum_{j=1}^{n}\left[\alpha\log\left(\sum_{i=1}^{K}\tilde{p}_i^{(1)}\phi_{ij}\right) + (1-\alpha)\log\left(\sum_{i=1}^{K}\tilde{p}_i^{(2)}\phi_{ij}\right)\right] =$$

$$= \alpha\sum_{j=1}^{n}\log\left(\sum_{i=1}^{K}\tilde{p}_i^{(1)}\phi_{ij}\right) + (1-\alpha)\sum_{j=1}^{n}\log\left(\sum_{i=1}^{K}\tilde{p}_i^{(2)}\phi_{ij}\right) =$$

$$= \alpha L(\boldsymbol{p}^{(1)}; \boldsymbol{x}) + (1-\alpha)L(\boldsymbol{p}^{(2)}; \boldsymbol{x}).$$

Справедливость соотношения (7) установлена.

Обратим внимание на то, что при доказательстве вогнутости функции $L(\boldsymbol{p}; \boldsymbol{x})$ никак не использовалась нормальность компонент смеси $\tilde{F}(x)$. Поэтому можно говорить о вогнутости любой логарифмической «сеточной» функции правдоподобия как функции весов.

Таким образом, задачу поиска оценок максимального правдоподобия для весов $\widetilde{p}_1, \ldots, \widetilde{p}_K$ можно свести к задаче условной оптимизации

$$J(\boldsymbol{p}) = L(\boldsymbol{p}; \boldsymbol{x}) \to \sup, \qquad \boldsymbol{p} \in \mathcal{P} \subset \mathbb{R}^K. \tag{9}$$

Заметим, что множество $\mathcal{P}$ - выпуклый компакт в $\mathbb{R}^K$, а функция $J(\boldsymbol{p})$ вогнута на $\mathcal{P}$. Из этого следует, что точка локального максимума $J(\boldsymbol{p})$ является точкой её глобального максимума на $\mathcal{P}$.

4 Алгоритм условного градиента

Вогнутость «сеточной» функции правдоподобия, установленная в разделе 3, дает возможность применения стандартных градиентных методов для поиска решения задачи(9). Это решение может быть численно найдено с помощью метода условного градиента, итерационный процесс для которого, как известно(см.[1]), имеет вид

$$\boldsymbol{p}^{(m+1)} = \boldsymbol{p}^{(m)} + \theta_m \cdot (\widehat{\boldsymbol{p}}^{(m)} - \boldsymbol{p}^{(m)}),$$

где $0 \le \theta_m \le 1$, а вспомогательное приближение $\widehat{\boldsymbol{p}}^{(m)}$ определяется как

$$\widehat{\boldsymbol{p}}^{(m)} = \arg\max_{\boldsymbol{p}\in\mathcal{P}} \langle \operatorname{grad} J(\boldsymbol{p}^{(m)}),\ \boldsymbol{p} - \boldsymbol{p}^{(m)} \rangle.$$

Здесь символ $\langle \boldsymbol{a},\ \boldsymbol{b} \rangle$ обозначает обычное скалярное произведение векторов $\boldsymbol{a}$ и $\boldsymbol{b}$ в $\mathbb{R}^K$.

Таким образом, на каждой (m-й) итерации необходимо решить две задачи. Во-первых, найти вспомогательное приближение $\widehat{\boldsymbol{p}}^{(m)}$ и, во-вторых, определить оптимальное значение параметра θ_m.

Несложно убедиться, что вектор $\widehat{\boldsymbol{p}}^{(m)}$, будучи решением задачи линейного программирования, находится в явном виде. С этой целью в дополнение к обозначениям, введенным в предыдущем разделе, обозначим

$$\psi_k^{(m)} = \sum_{j=1}^{n} \frac{\phi_{kj}}{\sum_{i=1}^{K} p_i^{(m)} \phi_{ij}}, \quad k = 1, \ldots, K.$$

Тогда

$$\langle \operatorname{grad} J(\boldsymbol{p}^{(m)}),\ \boldsymbol{p} - \boldsymbol{p}^{(m)} \rangle = \sum_{k=1}^{K} (p_k - p_k^{(m)}) \psi_k^{(m)} =$$

$$= \sum_{k=1}^{K} p_k \psi_k^{(m)} - \sum_{k=1}^{K} p_k^{(m)} \psi_k^{(m)}.$$

При этом от вектора $\boldsymbol{p} = (p_1, \ldots, p_K)$ зависит только первое слагаемое в правой части. Поэтому вектор $\widehat{\boldsymbol{p}}^{(m)}$, максимизирующий скалярное произведение $\langle \operatorname{grad} J(\boldsymbol{p}^{(m)}),\ \boldsymbol{p} - \boldsymbol{p}^{(m)} \rangle$, удовлетворяет соотношению

$$\widehat{\boldsymbol{p}}^{(m)} = \arg\max_{\boldsymbol{p}\in\mathcal{P}} \sum_{k=1}^{K} p_k \psi_k^{(m).}$$

Таким образом, если определить натуральное число $k^{(m)}$ как

$$k^{(m)} = \arg\max_{1\le k\le K} \psi_k^{(m)} = \arg\max_{1\le k\le K} \sum_{j=1}^{n} \frac{\phi_{kj}}{\sum_{i=1}^{K} p_i^{(m)} \phi_{ij}}$$

и положить

$$e_k^{(m)} = \begin{cases} 0, & k \ne k^{(m)}, \\ 1, & k = k^{(m)}, \end{cases} \quad k = 1, \ldots, K,$$

то несложно убедиться, что оптимальный вектор $\widehat{\boldsymbol{p}}^{(m)}$ будет иметь вид

$$\widehat{\boldsymbol{p}}^{(m)} = \boldsymbol{e}_{k^{(m)}} \equiv (e_1^{(m)}, \ldots, e_K^{(m)})^\top.$$

Теперь, чтобы определить оптимальное значение параметра θ_m, рассмотрим функцию

$$I_m(\theta) = J\big(\boldsymbol{p}^{(m)} + \theta \cdot (\widehat{\boldsymbol{p}}^{(m)} - \boldsymbol{p}^{(m)})\big) = J\big(\boldsymbol{p}^{(m)} + \theta \cdot (\boldsymbol{e}_{k^{(m)}} - \boldsymbol{p}^{(m)})\big) =$$

$$= \sum_{j=1}^{n} \log \left[\sum_{i=1}^{K} \phi_{ij} \left[p_i^{(m)} + \theta \cdot (e_i^{(m)} - p_i^{(m)}) \right] \right].$$

В качестве θ_m возьмем

$$\theta_m = \arg \max_{0 \le \theta \le 1} I_m(\theta). \tag{10}$$

Заметим, что

$$\frac{d^2 I_m(\theta)}{d\theta^2} = -\sum_{j=1}^{n} \left[\frac{\sum_{i=1}^{K} \phi_{ij}(e_i^{(m)} - p_i^{(m)})}{\sum_{i=1}^{K} \phi_{ij} \left[p_i^{(m)} + \theta \cdot (e_i^{(m)} - p_i^{(m)}) \right]} \right]^2 < 0.$$

Таким образом, функция

$$H_m(\theta) = I'_m(\theta) = \sum_{j=1}^{n} \frac{\sum_{i=1}^{K} \phi_{ij}(e_i^{(m)} - p_i^{(m)})}{\sum_{i=1}^{K} \phi_{ij} \left[p_i^{(m)} + \theta \cdot (e_i^{(m)} - p_i^{(m)}) \right]}$$

монотонно убывает на отрезке $[0, 1]$. Следовательно, уравнение

$$H_m(\theta) = 0 \tag{11}$$

имеет не больше одного корня. Значит, значение θ_m, являющееся корнем уравнения (11) и, стало быть, удовлетворяющее условию (10), единственно. Если функция $H_m(\theta)$ имеет значения одного знака на концах интервала $[0, 1]$, то решение задачи поиска θ_m тривиально: $\theta_m = 0$, если значение на концах отрицательно, $\theta_m = 1$, если положительно. В случае, когда функция $H_m(\theta)$ имеет значения разных знаков на концах интервала $[0, 1]$, задача поиска θ_m сводится к отысканию (единственного) корня уравнения (11). Для численного решения этого уравнения целесообразно использовать метод Риддера (см., например, [4]), дающий сходимость с точностью 10^{-13} в среднем за четыре итерации. Метод Ньютона при решении данной задачи может расходиться.

5 Применение алгоритма условного градиента для задачи разделения смесей. CPC-метод.

Напомним, что эффективный алгоритм решения задачи разделения смесей требуется, например, при использовании метода скользящего разделения смесей(CPC-метода) для работы с временными рядами(смотри[2]). Суть CPC-метода заключается в следующем. В анализируемом временном ряде выделяется «окно» - первые n компонент. Для данного «окна» решается задача разделения смесей. Далее, пока возможно, «окно» сдвигается влево, и для каждой позиции проводится оценка параметров компонент смеси. Полученная последовательность оценок позволяет наблюдать за динамикой изменения параметров смеси с течением времени

Рассмотрим особенности применения алгоритма условного градиента при решении задачи разделения смесей на примере анализа ряда из логарифмических приращений индекса CAC40 CPC-методом. В качестве анализируемых данных взяты минутные данные с 16 по 25 февраля 2009 года. Из исходных данных удалены наблюдения, относящиеся к приращениям за интервал, больший одной минуты. В рамках данной задачи необходимо провести оценку параметров смеси для 1257 положений «окна».

В качестве критерия останова итерационного процесса возьмем выполнение условия малости изменения значения целевой функции:

$$J\left(\boldsymbol{p}^{(m+1)}\right) - J\left(\boldsymbol{p}^{(m)}\right) < \epsilon.$$

Будем использовать два варианта выбора начального приближения. Первый - выбор стартовой точкой алгоритма вектора, компоненты которого равны, то есть

$$p_i = \frac{1}{K} \quad i = 1, \ldots, K.$$

Таблица 1: Результаты применения алгоритма условного градиента.

ϵ	1e-7		1e-8	
начальное приближение	одинак.	пред. итерация	одинак.	пред. итерация
среднее ч.и.	6472	2957	20603	12148
стандартное отклонение ч.и.	3562	2907	11264	10550
максимальное ч.и.	18918	15830	59874	55681
минимальное ч.и.	801	2	2615	3

Второй вариант - использование в качестве начального приближения вектора оценок, полученного для предыдущего положения «окна».

Как видно из Таблицы 1, скорость работы алгоритма для второго способа выбора стартовой точки выше почти в два раза.

CPC-метод имеет одну характерную черту, которая может быть проиллюстрирована на следующем примере. Рассмотрим выборку длинны M. Путь первые m элементов выборки - (независимые) реализации случайной величины $X_1 \sim \mathcal{N}(0, \sigma_1^2)$, а оставшиеся $(M - m)$ наблюдений относятся к $X_2 \sim \mathcal{N}(0, \sigma_2^2)$. Очевидно, что данный ряд получен из наблюдений дискретной смеси нормальных законов с одной компонентой(смешивающий закон вырожден в точке $\sigma = \sigma_1$ до момента m, и $\sigma = \sigma_2$ после него). Однако, при анализе выборки с помощью CPC-метода, можно получить результаты, не соответствующие реальности. Если «окно» содержит реализации случайных величин X_1 и X_2, то результатом решения задачи разделения смесей для данной позиции «окна» будет двухкомпонентная смесь. Если наблюдать за динамикой изменения оценок параметров, полученных CPC-методом, будет казаться, что какую-то часть времени наблюдаемая смесь состояла из двух компонент, что не соответствует действительности.

Похожую ситуацию можно наблюдать и на примере индекса CAC40. Известно(см., например,[5]), что интенсивность торгов на рынке во многом зависит и от неслучайных событий(например открытия бирж на других континентах влекут за собой резкие скачки интенсивности). Попробуем нормализовать выборку таким образом, чтобы средняя интенсивность в каждый момент времени оставалась одинаковой.

В качестве меры интенсивности торгов будем использовать выборочную дисперсию логарифмических приращений индекса. Процесс нормализации заключается в следующем:

1. По временному ряду, значительно большему исходного(в данном случае с 5 ноября 2008 по 6 марта 2009), вычислим выборочную дисперсию для каждой минуты торгов. Обозначим её σ_k^2, где $k = 1 \dots 510$ - номер минуты, к которой относится приращение.

2. Изменим значения полученного вектора заменив σ_k^2 на $\tilde{\sigma}_k^2$, где $\tilde{\sigma}_k^2$ - усредненная по 9 соседним значениям дисперсия для k-й минуты, поделенная на среднюю дисперсию за день.

3. Далее поделим каждое значение исходного ряда на соответствующее σ_k, $k = 1 \dots 510$. Таким образом, средняя интенсивность для каждой минуты торгов станет постоянной.

Сравним результаты применения алгоритма к двум рядам. На Рис.1 представлен результат применения CPC-метода к исходному ряду, на Рис.2 - к нормализованному. Видно, что на втором графике изменение параметров смеси с течением времени происходит более плавно, на графике присутствует непрерывные кривые, которые можно отнести к компонентам смеси, параметры которых непрерывно меняются с течением времени. Первый график содержит больше «рваных» кривых, его сложнее интерпретировать.

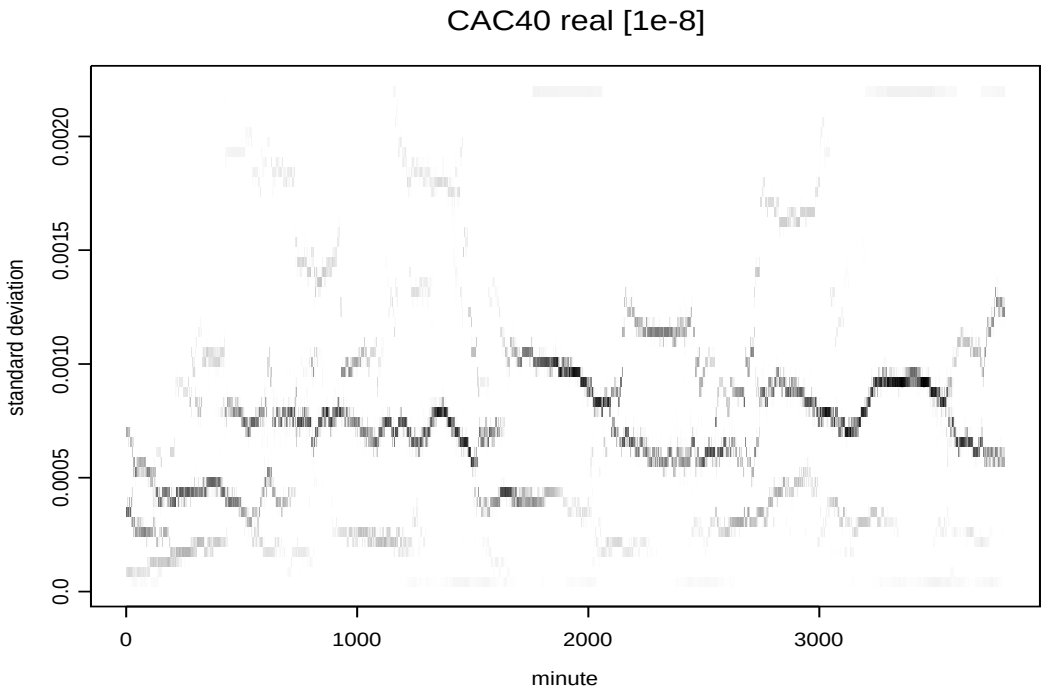


Рис. 1: Анализ логарифмических приращений индекса CAC40 с помощью CPC-метода, исходный ряд.

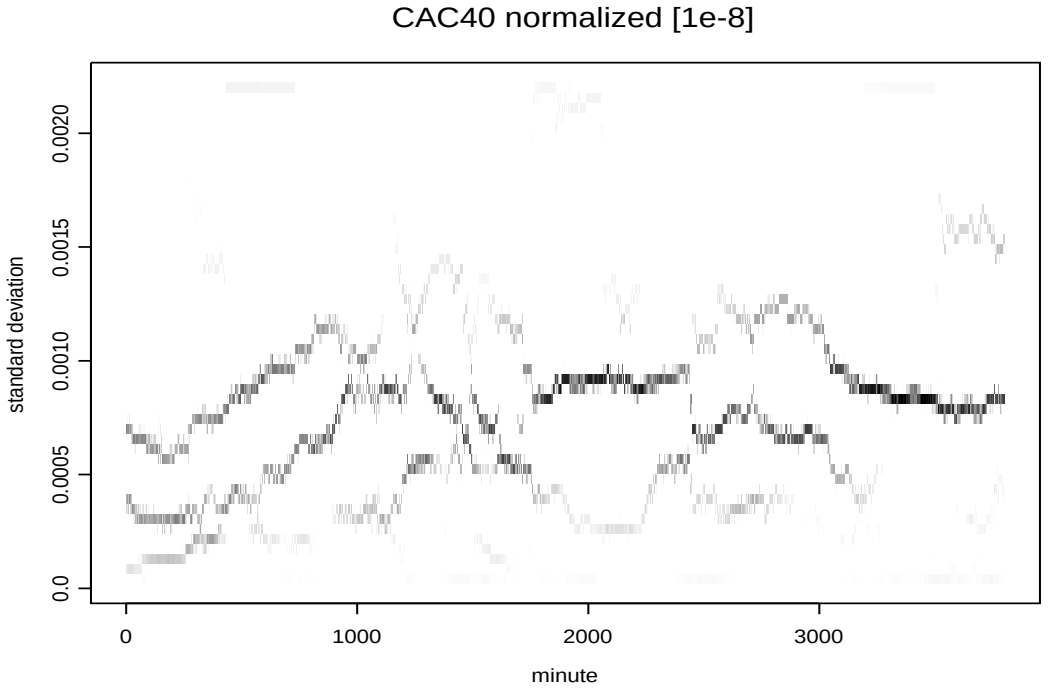


Рис. 2: Анализ логарифмических приращений индекса CAC40 с помощью CPC-метода, нормализованный ряд.

6 Заключение.

Сеточный метод максимального правдоподобия представляется подходящим для использования в скользящем режиме для решения задачи динамического разделения смесей. Возможность подстановки в качестве начального приближения результата, полученного для предыдущей позиции «окна» позволяет сильно увеличить быстродействие алгоритма. При этом скорость работы данного метода на порядок выше, чем у EM-алгоритма.

Список литературы

[1] Ф. П. Васильев. *Численные методы решения экстремальнях задач* Главная редакция физико-математической литературы изд-ва "Наука 1988.

[2] В. Ю. Королев. *Вероятностно-статистический анализ хаотических процессов с помощью смешанных гауссовских моделей. Декомпозиция волатильности финансовых индексов и турбулентной плазмы.* Изд-во ИПИРАН, Москва, 2007.

[3] В. Ю. Королев, Е. В. Непомнящий, А. Г. Рыбальченко и А. В. Виноградова. *Сеточные методы разделения смесей вероятностных распределений и их применение к декомпозиции волатильности финансовых индексов.* Информатика и ее применения, 2008, т. 2, вып. 2, с. 2–17.

[4] Ю. Ю. Тарасевич *Численные методы на Mathcad'е: Учебно-методическое пособие.* Астрахань: Астраханский гос. пед. ун-т, 2000.

[5] А. Н. Ширяев. *Основы стохастической финансовой математики. Том 1. Факты. Модели.* Москва: ФАЗИС, 1998.

УДК 519.688

МЕТОД ВЫДЕЛЕНИЯ В ТЕКСТЕ КОНСТРУКЦИЙ ПО ИХ ЛЕКСИКО-СИНТАКСИЧЕСКИМ ШАБЛОНАМ

© 2009 г. А. А. Носков

alexey.noskov@gmail.com

Кафедра Алгоритмических Языков

1 Введение

В настоящее время в различных областях компьютерной лингвистики и искусственного интеллекта существует потребность в средствах автоматизации выделения в тексте на естественном языке определенных языковых конструкций, в частности, согласованных именных словосочетаний (*усталое осеннее солнце*, *уходящий поезд*), глагольных групп (*шел по тротуару*, *писать стихи*), а также более сложных конструкций, характерных например для текстов научно-технического стиля (*под A будем понимать B*, *предположим, что C*) и т.п. До сих пор задача такого выделения обычно решалась каждый раз заново в условиях конкретного приложения по автоматической обработке текста, и для отдельных типов языковых конструкций. В данной работе предлагается новый метод, позволяющий осуществлять выделение в тексте достаточно широкого круга языковых конструкций, описанных в виде шаблонов языка LSPL[1].

В отличие от давно изучающейся задачи полного синтаксического анализа, при которой последовательно по предложениям распознается вся синтаксическая структура текста, решаемая нами задача предполагает распознавание в тексте только заданных конструкций по описанию их свойств. Безусловно, она может быть решена и на основе полного синтаксического анализа, однако полный анализ сам по себе является более сложной и высокозатратной задачей, избыточной для целей выделения только необходимых конструкций. Фактически, можно считать задачу выделения в тексте языковых конструкций, как разновидность поверхностного синтаксического анализа.

Если рассматривать возможность использования уже существующих инструментов для решения указанной задачи, то стоит отметить, что она быть решена достаточно просто при наличии подготовленных корпусов текстов, таких как НКРЯ (Национальный корпус русского языка) или ХАНКО (Хельсинский аннотированный корпус) [4] – за счет использования морфологической и синтаксической разметки текстов корпуса, подготовленной вручную лингвистами. Однако корпуса ограничены по набору текстов, в то время как часто возникает необходимость исследовать тексты, не включенные в корпус. Кроме того, в корпусах размечаются только те языковые конструкции, которые были признаны значимыми для задач, решаемых корпусом, в то время как нередко требуется выделять в текстах другие конструкции, для которых разметка в корпусе отсутствует.

К инструментам, позволяющим автоматизировать выделение нужных типов языковых конструкций, следует отнести такие системы для построения приложений обработки естественного языка, как GATE (General Architecture for Text Engineering) [5] и Ellogon [6]. Эти системы предлагают языковые средства для описания единиц текста (например, язык Jape в системе GATE) и программные средства для автоматического выделения фрагментов текста по такому описанию. Однако несмотря на универсальность этих систем их применение для русского языка представляет некоторые сложности. Большинство подобных систем разрабатывались для английского языка, который по ряду свойств отличается от русского языка, который является более флективным и имеет менее строгий порядок слов, что приводит к необходимости использовать при анализе русскоязычных текстов широкий набор морфологических характеристик слов. Поскольку указанные системы разрабатывались для английского языка, в них также нет специальных средств задания условия грамматического согласования, крайне важного для автоматического выделения конструкций русского языка.

Если же рассматривать отечественные системы, ориентированные на анализ текстов на русском языке, то среди них стоит отметить систему Alex [3], позволяющую описывать в виде лексических шаблонов слова, словосочетания и их сокращения, и затем автоматически выделять их в тексте. Основным недостатком системы является ограниченность языка шаблонов — в нем нет возможности использования морфологических характеристик слов описываемых языковых конструкций, что значительно сужает возможности системы.

В качестве более гибкого средства задания конструкций естественного языка для их автоматического выделения, был предложен язык LSPL [1], учитывающий особенности русского языка. Он позволяет в удобной форме специфицировать конструкции русского языка из достаточно широкого класса в виде так называемых лексико-синтаксических шаблонов, определяющих входящие в конструкцию слова с учетом их морфологических характеристик и задающих условия их грамматического согласования. При автоматическом выделении некоторой конструкции шаблон последовательно накладывается на текст, образуя так называемые варианты наложения, соответствующие различным случаям вхождения в текст этой конструкции. Рассматриваемый в настоящей работе метод выделения языковых конструкций в тексте разработан именно для случая их описания в виде LSPL-шаблонов.

2 Язык LSPL и задача выделения конструкций

Основным средством языка LSPL является шаблон, описывающий некоторую языковую конструкцию. Каждый шаблон в общем случае состоит из имени и набора альтернатив, описывающих различные варианты языковой конструкции. Например, если шаблон описывает пару слов, соединенных союзом «*и*» или «*а*», то он содержит две соответствующие альтернативы, разделенные вертикальной чертой: `N1 “и” N2 | N1 “а” N2`.

Элементами его альтернатив являются входящие в описываемую языковую конструкцию слова с их полностью или частично конкретизированными морфологическими характеристиками (часть речи, падеж, род, число и т.п.). При этом порядок следования элементов в шаблоне соответствует порядку элементов описываемой языковой конструкции. Например, шаблон `N V<t=past>` описывает конструкцию из существительного (`N`) и следующего за ним глагола (`V`) в прошедшем времени (`t=past`). Элемент шаблона может описывать любую из допустимых словоформ некоторого слова, в этом случае в шаблоне записывается начальная форма этого слова (в частности, для существительных – именительный падеж единственного числа), например: `A<лес>` описывает существительное «лес», находящееся в конструкции в любой из возможных форм.

Язык LSPL позволяет задавать условия грамматического согласования, указывающие равенство морфологических признаков у различных элементов-слов описываемой конструкции. Например, шаблон `A N <A.g=N.g, A.n=N.n>` описывает прилагательное со следующим за ним существительным, согласованные по роду (`g`) и числу (`n`).

Для описания конкретных строк, встречающихся в конструкции, используется запись вида `“строка”`. В частности, такой **элемент-строка** может быть использован для задания в шаблоне знаков пунктуации, например `“;”`.

Кроме указанных простых элементов шаблоны могут содержать и более сложные элементы, в частности, повторения, которые записываются в фигурных скобках. Например, запись `A<1,3> N <A=N>` обозначает последовательность, состоящую из одного, двух или трех прилагательных, за которыми следует согласованное с ними (по всем общим морфологическим характеристикам) существительное. Частным случаем повторения является опциональный элемент: к примеру, запись `[A] N <A=N>` задает в общем случае прилагательное вместе со согласованным с ним существительным, но прилагательное может быть опущено.

Достаточно часто удобно выделить некоторую стандартную конструкцию или ее часть в отдельный шаблон и дать ему имя, что допускается языком LSPL. Например, шаблон с именем `NameGroup`:

```
NameGroup = {A} N1 <A=N1>
```

описывает именную группу, состоящую из последовательности прилагательных и согласованного с ними существительного (например, *белый снег* или *теплый летний дождь*, но не *белый снега*). После такого описания можно использовать этот шаблон в других шаблонах, к примеру шаблон `NameGroup V` задает последовательность из такой именной группы и следующего за ней глагола (*пушистый кот спал*, но не *пушистый кошка спала*).

LSPL-шаблон может иметь параметры, задающие морфологические характеристики описываемой шаблоном конструкции. Например, запись `NG = A N (N.c, N.g)` — шаблон с именем `NG`, параметрами которого являются падеж (`c`) и род (`g`) входящего в шаблон существительного. Для сокращения может быть использована запись вида `NG = A N (N)` — в таком шаблоне параметрами являются все морфологические характеристики существительного. Параметры шаблона особенно ценны при использовании шаблонов в других шаблонах. Например, можно описать именную группу как:

```
NG = {A} N1 <A=N1> [N2<c=gen>] (N1)
```

Такая именная группа задает последовательность прилагательных с согласованным с ними существительным и опциональным существительным в родительном падеже (к примеру, *белые шапки гор*). В дальнейшем при использовании такого шаблона можно задать условие согласования: конструкция `NG1 V <NG1=V>` соответствует именной группе `NG`, согласованной по всем морфологическим характеристикам с глаголом `V` и описывает словосочетания вида *белый кот спал*, но не *белый кот спала* (последнее не согласовано).

Кроме использования уже описанных шаблонов в других шаблонах язык LSPL позволяет строить рекурсивные определения шаблонов:

```
NG = {A} N1 <A=N1> [NG2<c=gen>] (N1)
```

Этот шаблон именной группы отличается от предыдущего тем, что опциональная часть может быть представлена не только одиночным существительным, но и произвольной именной группой. К примеру, этот шаблон описывает такие конструкции, как *тоненькая струйка дыма далекого пожара*.

Таким образом, язык LSPL является достаточно мощным средством для описания различных языковых конструкций. Для заданного текста и LSPL-шаблона задача выделения в тексте на русском языке конструкций по шаблону может быть уточнена так: найти и выделить все фрагменты текста, представляющие собой языковую конструкцию, свойства которой описаны в виде LSPL-шаблона. При выделении этой конструкции шаблоны накладываются на текст, образуя так называемые **варианты наложения** – отрезки текста (непрерывные последовательности символов текста), соответствующие выделенной конструкции.

LSPL-шаблон может содержать входящие в описываемую языковую конструкцию знаки пунктуации (в виде элементов-строк), однако конструкции могут быть описаны и без учета пунктуации. Очевидно, что в последнем случае при выделении конструкций пунктуацию необходимо просто игнорировать. Для поддержки обоих возможностей в предлагаемом методе вводится два режима выделения, которые в дальнейшем будут обозначаться «с учетом пунктуации» и «без учета пунктуации».

Отрезок текста, представляющий вариант наложения LSPL-шаблона, включает подотрезки, соответствующие простым элементам этого шаблона, например, словам или знакам пунктуации (в случае анализа с учетом пунктуации). Кроме них, в общем случае в отрезке есть символы, незначащие с точки зрения производимого анализа, например, пробельные и управляющие символы, а также знаки пунктуации в случае выделения без учета пунктуации. В целом, рассматриваемый отрезок разбивается на непересекающиеся отрезки: **значимые** и **незначимые** для проводимого анализа, причем такое деление отрезка на непересекающиеся подотрезки зависит только от режима выделения, но не от конкретного шаблона, и его можно распространить на весь текст.

Например, при анализе без учета пунктуации текст «*Человек шел, оглядываясь назад.*» будет содержать два незначимых отрезка, соответствующих пробельным символам, один незна-

чимый отрезок, включающий запятую и пробел за ней, и один незначимый отрезок, соответствующий точке в конце предложения (см. Рис. 1). Если же пунктуация учитывается, то незначимыми отрезками будут только те, которые состоят из пробельных символов.

Указанное разбиение текста на непересекающиеся отрезки используется для построения внутреннего представления текста.

Рис. 1: Разбиение текста на отрезки

3 Внутренние структуры данных

3.1 Представление текста

Для организации эффективного выделения языковых конструкций было предложено представление текста в виде графа, которым оперирует алгоритм выделения. Узлы этого графа соответствуют незначимым отрезкам текста, а ребрами являются различные синтаксические интерпретации лежащих между ними значимых отрезков текста. Под синтаксической интерпретацией отрезка текста будем понимать набор значений морфологических характеристик слов, входящих в этот отрезок. Такое представление текста в виде графа позволяет при наложении шаблонов рассматривать различные сочетания интерпретаций входящих в текст слов и словосочетаний.

При построении внутреннего представления текста сначала осуществляется его разбиение на значимые и незначимые отрезки, при этом идущие подряд незначимые отрезки склеиваются и образуют вершины графа. Построенные вершины нумеруются, начиная с 0, в направлении от начала к концу текста. Затем выполняется морфологический анализ слов, входящих в значимые отрезки и построение ребер графа между соседними вершинами; ребра соответствуют установленным в ходе анализа синтаксическим (морфологическим) интерпретациям этих слов. Граф считается ориентированным: все ребра направлены в сторону вершин с большим номером. Кроме того, такой граф не содержит ориентированных циклов.

Любая пара вершин в этом графе однозначно определяет некоторый отрезок текста, фиксируя его начало и конец, а различные пути в графе между этими вершинами описывают возможные сочетания синтаксических интерпретаций входящих в этот отрезок значимых подотрезков. Таким образом, построение графа текста позволяет в дальнейшем рассматривать задачу выделения конструкций в тексте, как поиск путей в графе, удовлетворяющих синтаксическим требованиям, накладываемым шаблоном.

На Рис. 2 приведен пример графа текста, построенного для режима с учетом пунктуации.

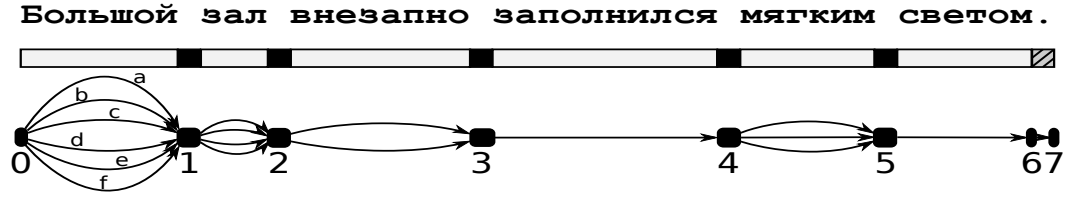


Рис. 2: Граф текста

В общем случае между парой соседних вершин проходит несколько ребер — они соответствуют различным синтаксическим (морфологическим) интерпретациям слов в соответствующих отрезках текста. Как видно из рисунка, количество синтаксических интерпретаций может оказаться достаточно велико. Например, у слова «*большой*» их 6, представляющих различные сочетания морфологических характеристик: a – мужской род, именительный падеж; b – мужской род, винительный падеж; c – женский род, родительный падеж; d – женский род, дательный падеж; e – женский род, творительный падеж; f – женский род, предложный падеж.

3.2 Представление выделенных конструкций

Любая обнаруженная в тексте в результате наложения шаблона конструкция будет представляться в этом же графе текста дополнительным ребром, соединяющими вершины — конечные точки соответствующего отрезка текста. Поскольку накладываемые шаблоны конструкций могут иметь параметры, в качестве которых выступают морфологические характеристики входящих в них элементов, то и дополнительным ребрам приписываются значения этих параметров (т.е. значения морфологических характеристик элементов конструкции). В общем случае, одному и тому же отрезку может соответствовать несколько различных вариантов наложения, отличающихся только значениями морфологических характеристик. Пример представления вариантов наложения шаблона `A N <A=N>(N)` (описывающего согласованную пару из прилагательного и существительного) в графе текста приведен на Рис. 3.

Рис. 3: Результаты наложения шаблона `A N <A=N>(N)`

На рисунке жирно выделены два ребра между вершинами 0 и 2, соответствующих двум вариантам наложения этого шаблона на отрезок текста «*большой зал*» (двум синтаксическим интерпретациям этого отрезка) и одно ребро между вершинами 4 и 6, соответствующее наложению этого шаблона на отрезок «*мягким светом*».

Рассмотренное представление в графе выделенных конструкций позволяет единообразно обрабатывать как элементы-слова, так и экземпляры шаблонов и не производить повторно выделение конструкций, соответствующих уже наложенным шаблонам.

3.3 Индексы

Для увеличения производительности наложения шаблонов одновременно с построением графа текста строится набор индексов – структур данных, позволяющих быстро определять множество ребер графа, с которых есть смысл начинать наложение шаблона. Каждый индекс решает следующую задачу: по заданному ключу получить упорядоченный по номеру начальной вершины список ребер графа, обладающих заданными свойствами. В ходе выделения языковых конструкций используется три типа индексов:

- Индекс частей речи – ключом для этого индекса является часть речи и в качестве результата выдается множество ребер графа, представляющих все найденные в тексте словоформы заданной части речи. Индекс позволяет оптимизировать наложение шаблона в достаточно распространенной ситуации, когда шаблон начинается с элемента-слова, имеющего конкретную часть речи.

- Индекс шаблонов – ключом является некоторый уже наложенный шаблон и по нему извлекается множество всех вариантов его наложения. Таким образом оптимизируется наложение шаблона в случае, когда его первым элементом является другой, уже наложенный шаблон.
- Индекс слов – ключом является начальная форма слова, а в качестве результата выдается множество всех ребер, представляющих его словоформы в тексте. Этот индекс используется при наложении шаблонов, которые начинаются с конкретных слов (как, например, в шаблоне `A<красный>`).

На Рис. 4 изображены индексы частей речи (существительное, наречие и глагол) и индекс шаблона `A N<A=N> (N)`: индексы ссылаются на соответствующие ребра графа текста «*Большой зал внезапно заполнился мягким светом*».

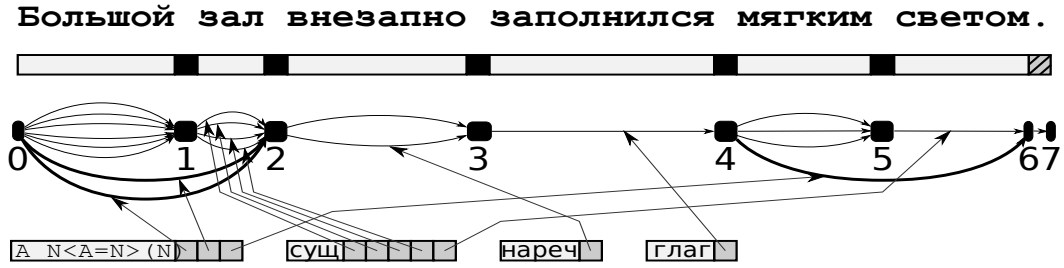


Рис. 4: Индексы в графе текста

3.4 Представление шаблонов

LSPL-шаблоны, описывающие выделяемую в тексте конструкцию, также представляются в специальной форме. В общем случае, каждый шаблон состоит из набора альтернатив, каждая из которых представляет из себя последовательность элементов шаблона.

Во внутреннем представлении каждый элемент шаблона имеет набор контекстных условий, описывающих ограничения на его наложение. Этими условиями являются различные ограничения на морфологические характеристики элементов (в частности, конкретизация падежа, рода, числа и т.п.), а также условия грамматического согласования. Однако, если в самом шаблоне они обычно расположены в конце, то при переводе шаблона во внутреннее представление условия согласования сдвигаются максимально влево, до позиции последнего элемента, участвующего в согласовании. При наложении шаблона это позволяет более эффективно отбрасывать несогласованные части конструкции.

4 Алгоритм наложения шаблона

Наложение шаблона на текст рассматривается как поиск в графе текста и включает 3 основных этапа:

I Определение начальных ребер – множества ребер, с которых может быть начат поиск в графе;

II Поиск путей в графе, соответствующих шаблону, начиная с полученных на предыдущем этапе ребер;

III Группировка вариантов наложения – найденные пути группируются и образуют варианты наложения, добавляемые в соответствующий индекс и граф в виде новых ребер.

Рассмотрим каждый этап детальнее.

4.1 Определение начальных ребер

Цель этого этапа – максимально сузить множество ребер, с которых могут начинаться языковые конструкции, выделенные по шаблону. Для этого предлагается использовать следующую схему работы этапа:

1. Для каждой альтернативы шаблона определить множество начальных элементов-слов или экземпляров других шаблонов;

2. Для каждого элемента этого множества по соответствующим индексам извлекаются начальные ребра графа, которые могут соответствовать этому элементу. В частности, для элемента-слова извлекается множество всех ребер, представляющих слова той же части речи, для экземпляра шаблона – множество всех ребер, представляющих варианты наложения этого шаблона. Если для какого-то элемента шаблона необходимых данных в индексе не находится (например, используемый шаблон еще не был наложен), то выполнение этапа заканчивается и следующий этап алгоритма (поиск путей в графе) осуществляется в предположении, что шаблон может начинаться с любого ребра;

Например, при наложении шаблона `A N<A=N>(N)` работа по указанной схеме происходит следующим образом:

1. Шаблон содержит одну альтернативу, множество начальных элементов состоит из одного элемента-слова – `A` (прилагательное, без каких-либо конкретизированных характеристик);

2. Начальные ребра определяются на основе индекса частей речи (индекс прилагательных) и формируется множество всех ребер, представляющих в тексте прилагательные в их конкретных синтаксических интепретациях; на Рис. 5 для рассматриваемого примера эти ребра выделены жирно.

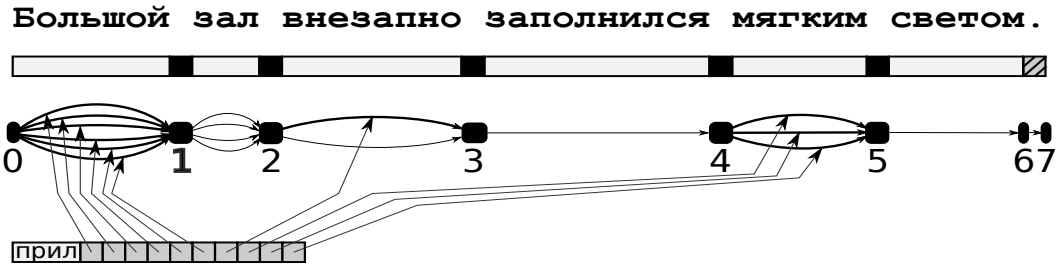


Рис. 5: Начальные ребра

Дальнейший поиск вариантов наложения в графе проводится последовательно, начиная с каждого выделенного ребра, представляющего прилагательное в тексте: шести ребер, начинающихся с узла 0 (различные синтаксические интепретации прилагательного «*большой*»), одного ребра, начинающегося с узла 2 (прилагательное «*внезапный*» в краткой форме) и трех ребер, начинающихся с узла 4 (прилагательное «*мягкий*»).

4.2 Поиск путей в графе

На этом этапе рассматривается множество всех путей в графе, начинающихся с заданного ребра. В этом множестве ищутся те пути, которые соответствуют последовательности элементов шаблона. Поиск нужных путей представляет из себя обход графа в глубину с откатом назад, при этом на каждом шаге проверяются все допустимые продолжения пути, которые соответствуют текущему элементу шаблона (например, слову определенной части речи или

варианту наложения заданного шаблона) и его контекстным условиям (например, конкретизированные падеж, род и другие морфологические характеристики).

Множество рассматриваемых при поиске путей образует так называемое дерево поиска, которое получается склеиванием путей графа с одинаковым префиксом. Дерево поиска отражает процесс перебора ребер графа при поиске пути. Например, для шаблона `A N<A=N>(N)` обход графа осуществляется по путям, показанным на Рис. 6.

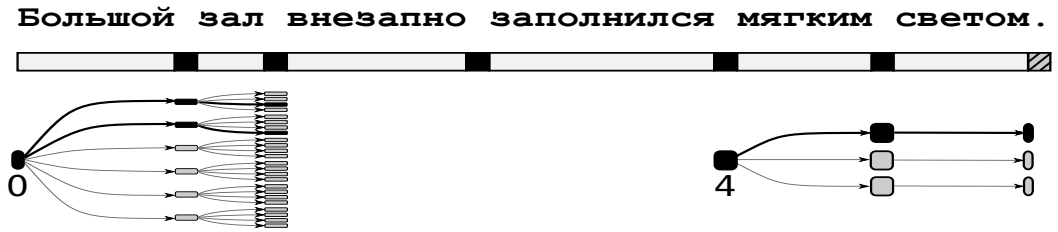


Рис. 6: Деревья поиска шаблона `A N<A=N>(N)`

На рисунке жирным выделены пути в деревьях, приводящие к успешному наложению шаблона: это 3 пути, удовлетворяющих условию грамматического согласования из 27 возможных в графе путей, соответствующих последовательности элементов накладываемого шаблона. Таким образом, условия согласования позволяют отбросить значительное количество результатов.

Перед описанием третьего этапа рассмотрим подробнее два важных момента на этапе поиска путей в графе – обработку условий согласования и сложных конструкций.

4.3 Обработка условий согласования и сложных конструкций

Для проверки условий согласования в процессе наложении шаблона необходимо сравнивать характеристики различных элементов. Один из способов состоит в том, чтобы проверять условия согласования после того, как вся последовательность ребер графа, образующих вариант наложения, построена. Однако значительно более эффективной является проверка условий согласования сразу, как только становятся конкретизированы все входящие в условие морфологические характеристики.

Для этого в процессе поиска в графе поддерживается так называемый контекст наложения, отражающий состояние процесса наложения шаблона. Как только какой-либо элемент шаблона ставится в соответствие ребру графа, в контекст наложения добавляется пара «элемент — ребро». Впоследствии, если встречается условие согласования, то ребра, соответствующие участвующим в согласовании элементам, извлекаются из контекста и их характеристики проверяются на предмет выполнения условия. Поскольку во внутреннем представлении шаблонов условия согласования максимально сдвинуты влево, такой способ обеспечивает более раннее отсечение несогласованных вариантов наложения.

На Рис. 7 приведено дерево поиска вариантов наложения шаблона `A N Av V <A=N,N=V,Av=V>` (прилагательное `A` согласовано с существительным `N`, существительное – с глаголом `V`, а глагол – с наречием `Av`) на текст «*Большой зал внезапно заполнился мягким светом*»

Жирными линиями на рисунке отмечены два пути, приводящие к успешному наложению. Если осуществлять проверку согласования после рассмотрения всех элементов шаблона, то это привело бы к рассмотрению всех путей, изображенных на рисунке. Однако то, что согласование `A=N` может быть проверено сразу после обработки отрезка текста «*зал*» позволяет обрывать пути в дереве поиска, рассматривая только те, которые выделены жирным на рисунке. Как видно, это приводит к значительному сокращению множества рассматриваемых путей.

Сложные элементы шаблона, т.е. повторения (например, `{A}`) и экземпляры шаблонов (например, использование шаблона `NG` в шаблоне `NG V`) могут быть поставлены в соответствие

Рис. 7: Дерево поиска шаблона `A N Av V <A=N,N=V,Av=V>`

последовательности из нескольких ребер, в отличие от простых элементов, которые накладываются на одно ребро. Для того, чтобы не нарушать единого принципа представления информации о наложениях (каждому элементу шаблона ставится в соответствие одно ребро), процесс наложения сложных элементов включает добавление специального группирующего ребра над всей последовательностью, соответствующей сложному элементу.

При обработке экземпляра шаблона возможна ситуация, когда соответствующий шаблон еще не был наложен и, следовательно, его варианты наложения отсутствуют в графе текста – в этом случае с вершины графа, на которой остановился процесс поиска, начинается процесс наложения этого шаблона.

4.4 Группировка вариантов наложения

В общем случае каждый значимый отрезок в тексте имеет несколько синтаксических интерпретаций, и, следовательно, представляется несколькими кратными ребрами в графе. Поскольку при наложении шаблона выполняется поиск всех допустимых путей в графе, то наличие в графе кратных ребер означает существование различных путей между двумя вершинами, и, значит, возможны различные варианты наложения шаблона на один и тот же отрезок текста.

Количество вариантов наложения очень быстро растет с увеличением длины и сложности шаблона, что приводит к увеличению расхода памяти на поддержание графа и значительному снижению эффективности поиска. Кроме того, для человека описанная ситуация выглядит как множество практически неотличимых друг от друга вариантов наложений шаблона на один и тот же отрезок текста.

Для того, чтобы избежать подобных проблем в ходе поиска, в графе используется информация о параметрах шаблона: предполагается, что те морфологические характеристики, которые не вошли в параметры шаблона, не являются важными (в том смысле, что могут быть опущены при дальнейшем анализе результатов человеком и наложении других шаблонов). Например, для шаблона `A N <A=N>(N)` важными считаются только морфологические характеристики существительного (но не прилагательного). Это позволяет осуществить группировку вариантов наложения по различным наборам значений параметров шаблона, т.е. заменить одним вариантом все варианты наложения, различающиеся только значениями характеристик, не являющихся важными, что приводит к значительному уменьшению количества рассматриваемых в дальнейшем вариантов наложения.

Например, шаблон `A N<A=N>(N)` имеет 2 варианта наложения на отрезке «*Большой зал*» (см. Рис. 6), соответствующие именительному и винительному падежу слова «зал» (результаты сгруппированы только по синтаксическим интепретациям прилагательного), а шаблон `A N <A=N>`, отличающийся только тем, что не имеет параметров после группировки (по синтаксическим интепретациям как прилагательного, так и существительного) имеет уже только один вариант наложения на тот же отрезок.

5 Заключение

Описанный метод был реализован в программном комплексе, предназначенном для анализа русскоязычных текстов с использованием лексико-синтаксических шаблонов. Реализованный программный комплекс состоит из четырех компонентов:

- Ядро, написанное на языке C++ и реализующее предложенный метод;
- Набор консольных утилит, реализованных на языке C++, предназначенных для автоматического анализа текста и интеграции ядра с различными скриптами;
- Прикладной интерфейс для языка Java, позволяющий интегрировать ядро в приложения обработки естественного языка, разработанные на языке Java;
- Графический пользовательский интерфейс для анализа текста лингвистом.

Комплекс был успешно протестирован в задаче распознавания регулярных конструкций русского языка [2] и в настоящее время используется для решения задачи терминологического анализа русскоязычного текста.

Представленный метод позволяет осуществлять выделение в тексте на русском языке различных языковых конструкций, описанных в виде LSPL-шаблонов, и применим в различных областях обработки текста на естественном языке.

Список литературы

[1] Большакова Е.И., Баева Н.В., Бордаченкова Е.А., Васильева Н.Э., Морозов С.С. *Лексико-синтаксические шаблоны в задачах автоматической обработки текстов.* Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог'2007. М.: Издательский центр РГГУ, 2007, с. 70-75.

[2] Большакова Е.И., Васильева Н.Э. *Формализация лексико-синтаксической информации для распознавания регулярных конструкций естественного языка.* Программные продукты и системы, 2008, № 4, с. 103 - 106.

[3] Жигалов В.А., Жигалов Д.В., Жуков А.А., Кононенко И.С., Соколова Е.Г., Толдова С.Ю. *Система Alex как средство для многоцелевой автоматизированной обработки текстов.* Труды международного семинара Диалог'2002 «Компьютерная лингвистика и интеллектуальные технологии». М.: Наука, 2002. Т.2, с. 192-208.

[4] Резникова Т.И., Копотев М.В. *Лингвистически аннотированные корпуса русского языка (обзор общедоступных ресурсов)* Национальный корпус русского языка: 2003—2005. М.: Индрик, 2005.

[5] Bontcheva K., Cunningham H., Maynard, D., Tablan, V., and Saggion, H. *Developing Reusable and Robust Language Processing Components for Information Systems using GATE.* In: Proceedings of the 13th international Workshop on Database and Expert Systems Applications, DEXA. IEEE Computer Society, Washington, DC. 2002, pp. 223-227.

[6] Petasis G., Karkaletsis V., Paliouras G., Androutsopoulos I. and Spyropoulos C. D. *Ellogon: A New Text Engineering Platform.* In: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002). Las Palmas, Canary Islands, 2002, pp. 72-78.

УДК 517.977.5

OPTIMAL CONTROL IN THE SIMPLEST INVESTMENT ALLOCATION MODEL WITH INFINITE TIME HORIZON

© 2009 г. A. I. Puchkova

apuchkova@gmail.com

Optimal control department

1 Introduction. Dmitruk model

Assume that at each moment of time economic productivity Y can be divided into consumption C and investments I:

$$Y(t) = C(t) + I(t) = (1 - u(t))Y(t) + u(t)Y(t) \ .$$

The portion allocated to investments is denoted as a dimensionless allocation variable $u(t) \in [0, 1]$, treated as a function of time t. The main purpose of this paper is to examine the optimal regime for this allocation variable.

Investments are used both for capital increase and for renewal of depreciated capital. Let $K(t)$ be an amount of capital at time t. Then $\dot{K}(t)$ is the capital growth rate. It is required to renew $\mu K(t)$ amount of capital, where μ is an amortization coefficient. The following dynamics of K is assumed:

$$I(t) = \dot{K}(t) + \mu K(t) \tag{1}$$

where the reciprocal of the depreciation rate has been used as the unit of time. Suppose that economic productivity can be expressed by a production function of the following form:

$$Y(t) = f(K(t), L(t))$$

i. e. Y depends on capital K and labour resources L. It is assumed that the total amount of labour resources remains constant, i. e. $L(t) \equiv L$. Then

$$I(t) = u(t)Y(t) = u(t)f(K(t), L) \ . \tag{2}$$

For the sake of simplicity we suppose that $\mu = 1$. Combining equations (1) and (2) we obtain:

$$\dot{K}(t) = u(t)f(K(t), L) - K(t) \quad \text{or} \quad \frac{\dot{K}(t)}{L} = u(t)\frac{f(K(t), L)}{L} - \frac{K(t)}{L} \ . \tag{3}$$

Let $x(t)$ denote the average amount of capital per labourer:

$$x(t) = \frac{K(t)}{L}.$$

From economic considerations, see [4], the production function $f(K, L)$ is assumed to be first-order homogeneous:

$$f(\alpha K, \alpha L) = \alpha f(K, L) \quad \forall \alpha > 0 \ .$$

For $\alpha = 1/L$ we have

$$\frac{f(K(t), L)}{L} = f\left(\frac{K(t)}{L}, 1\right) = f\left(x(t), 1\right).$$

The function $f\left(x, 1\right)$ will be denoted as a one-variable function $g(x)$. This function $g(x)$ measures productivity per labourer, which depends on x. Equation (3) may be written as

$$\dot{x} = -x + g(x)\ u.$$

If $g(x) = x^{\varepsilon}$ and ε is small enough, then we can consider $g(x) \approx 1$. Therefore

$$\dot{x}(t) = -x(t) + u(t).$$

Classical economic theory treats optimality in terms of maximal utility. Accordingly, the following integral is to be maximized:

$$\int_0^{+\infty} e^{-\rho t} U(x(t))dt \ \to \max_{u(\cdot)}$$

where $U(x)$ is a utility function and ρ is a positive discount coefficient. Alternatively, with

$$F(x(t)) = -U(x(t))$$

we obtain the following minimization problem:

$$\int_0^{+\infty} e^{-\rho t} F(x(t))dt \ \to \min_{u(\cdot)}.$$

We are thus led to consider the following optimal control problem at infinite time horizon:

$$\begin{cases} \dot{x} = -x + u, \quad 0 \le t < +\infty \\ x(0) = x_0 \\ J = \int_0^{+\infty} e^{-\rho t} \ F(x(t))dt \ \to \min_{u(\cdot)} \\ 0 \le u(t) \le 1 \end{cases}. \tag{4}$$

Here x and u are one-dimensional phase variable and control respectively, ρ and x_0 are positive constants, and $F(x)$ is a continuous function, defined for all $x \in R$, such that F is decreasing for $x < a$ and increasing for $x > a$ with a unique minimum at point $a > 0$. We do not suppose that $F(x)$ is smooth. The admissible control set consists of piecewise continuous functions $u(t)$, defined for $t \in [0, +\infty)$, with values $\in [0, 1]$, such that the improper integral J is convergent. Problem (4) was suggested by Professor A. V. Dmitruk.

2 Preliminary results and investigations

Lemma 1 *For any admissible pair $(u(t), x(t))$ of problem (4) the following double inequality is valid:*

$$0 < x^-(t) \le x(t) \le x^+(t) \quad \forall t \ge 0$$

where $x^-(t) = x_0 e^{-t}$ is the trajectory corresponding to $u \equiv 0$; and $x^+(t) = (x_0 - 1)e^{-t} + 1$ is the trajectory corresponding to $u \equiv 1$.

Proof. Let us consider the system:

$$\begin{cases} \dot{x} = -x + u(t) \\ x(0) = x_0 \end{cases}$$

with the following solution:

$$x(t) = e^{-t} \left(x_0 + \int_0^t e^s u(s)ds \right).$$

Using $0 \le u \le 1$ we find:

$$x(t) \ge e^{-t} x_0 = x^-(t)$$

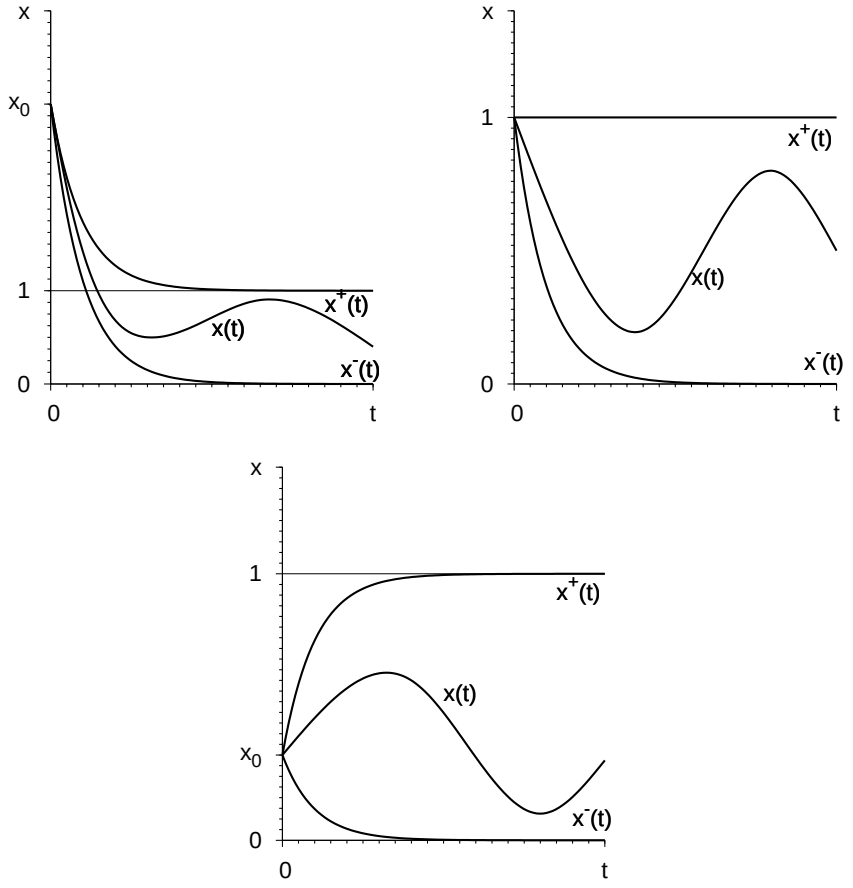


Figure 1: possible trajectories $x(t)$ for $x_0 > 1$, $x_0 = 1$ and $x_0 < 1$ correspondingly.

$$x(t) \le e^{-t}\left(x_0 + \int\limits_0^t e^s ds\right) = e^{-t}(x_0 - 1) + 1 = x^+(t)\ .$$

Lemma 1 is proved.

The solution of problem (4) depends on whether the controlled system is in the state $(u = a, x = a)$. First we consider the case $0 < a \le 1$. In this case the singular regime $(u(t) \equiv a, x(t) \equiv a)$ is possible. The value of the singular control a belongs to control domain $[0, 1]$.

Theorem 1 *For $0 < a \le 1$ the optimal solution $(u_*(t), x_*(t))$ for problem (4) has the following form:*

1. *in case $0 < x_0 < a < 1$:*

$$u_*(t) = \begin{cases} 1, & 0 \le t < \tau \\ a, & \tau \le t < +\infty \end{cases}$$
$$x_*(t) = \begin{cases} x^+(t), & 0 \le t < \tau \\ a, & \tau \le t < +\infty \end{cases}$$

where $\tau = \ln\left(\dfrac{1 - x_0}{1 - a}\right) > 0$;

2. *in case $0 < x_0 < a = 1$:*

$$u_*(t) \equiv 1, \quad x_*(t) \equiv x^+(t) \quad \forall t \ge 0\ ;$$

3. in case $x_0 = a$*:*

$$u_*(t) \equiv a, \quad x_*(t) \equiv a \quad \forall t \geq 0\ ;$$

4. in case $x_0 > a$*:*

$$u_*(t) = \begin{cases} 0, & 0 \leq t < \tau \\ a, & \tau \leq t < +\infty \end{cases}$$
$$x_*(t) = \begin{cases} x^-(t), & 0 \leq t < \tau \\ a, & \tau \leq t < +\infty \end{cases}$$

where $\tau = \ln\left(\dfrac{x_0}{a}\right) > 0$.

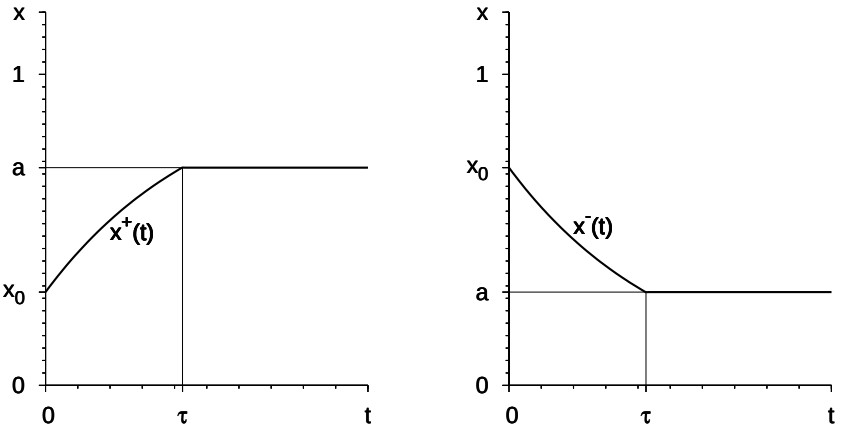


Figure 2: $x_*(t)$ for $0 < x_0 < a < 1$ and $x_0 > a$, respectively.

Proof. Find the increment of the functional

$$\Delta J = J[\bar{u}] - J[u_*]$$

for any admissible pair

$$\bar{u}(t), \bar{x}(t), 0 \leq t < +\infty.$$

1.

$$\Delta J = \int_0^{+\infty} e^{-\rho t} F(\bar{x}(t))dt - \int_0^{+\infty} e^{-\rho t} F(x_*(t))dt =$$

$$= \int_0^{\tau} e^{-\rho t} \left(F(\bar{x}(t)) - F(x^+(t))\right) dt + \int_{\tau}^{+\infty} e^{-\rho t} \left(F(\bar{x}(t)) - F(a)\right) dt.$$

$\bar{x}(t) \leq x^+(t) \leq a$ for $t \in [0, \tau]$ and $F(x)$ is decreasing for $x < a$, so $F(\bar{x}(t)) \geq F(x^+(t))$ for $t \in [0, \tau]$. $F(\bar{x}(t)) \geq F(a)$, since a is the minimum point of $F(x)$. Both integrands are nonnegative. Therefore, $\Delta J \geq 0$.

2.

$$\Delta J = \int_0^{+\infty} e^{-\rho t} \left(F(\bar{x}(t)) - F(x^+(t))\right) dt.$$

In this case $F(\bar{x}(t)) \geq F(x^+(t))$, due to $\bar{x}(t) \leq x^+(t) \leq a = 1$ and $F(x)$ decreases for $x < a = 1$. Thus $\Delta J \geq 0$.

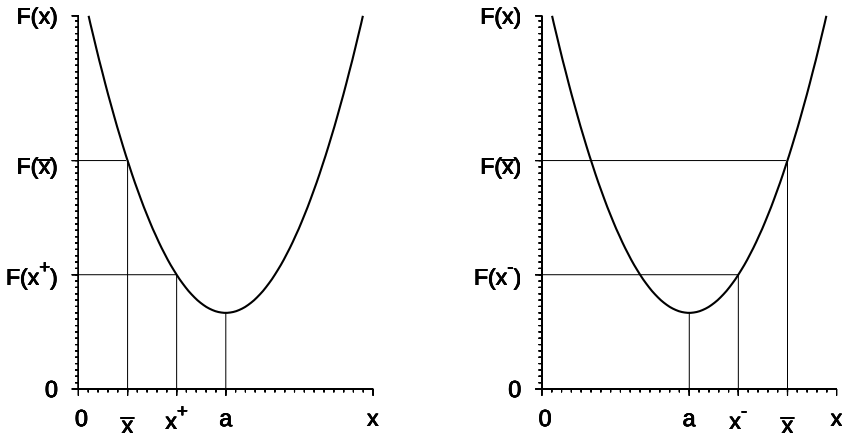


Figure 3: $F(x)$ for $0 < x_0 < a < 1$ and $x_0 > a$ respectively.

3.

$$\Delta J = \int_0^{+\infty} e^{-\rho t} \left(F(\bar{x}(t)) - F(a)\right) dt \geq 0.$$

4.

$$\Delta J = \int_0^{\tau} e^{-\rho t} \left(F(\bar{x}(t)) - F(x^-(t))\right) dt + \int_{\tau}^{+\infty} e^{-\rho t} \left(F(\bar{x}(t)) - F(a)\right) dt \geq 0.$$

$F(\bar{x}(t)) \geq F(x^-(t))$ for $t \in [0, \tau]$, since $a \leq x^-(t) \leq \bar{x}(t)$ for $t \in [0, \tau]$ and $F(x)$ is increasing for $x > a$. In all cases $\Delta J \geq 0$, i.e. the pair $(u_*(t), x_*(t))$ is optimal. This completes the proof.

This theorem can be interpreted as follows. For $0 < a \leq 1$ the singular control regime $u(t) \equiv a$ is more profitable. Thus it is reasonable that the optimal trajectory $x(t)$ is such that it attains the value a as quickly as possible and thereafter remains at this value.

Next, we will study the problem for $a > 1$. In this case singular control does not arise.

Theorem 2 *For $0 < x_0 \leq a$, $a > 1$ the optimal solution $(u_*(t), x_*(t))$ for problem (4) has the following form:*

$$u_*(t) \equiv 1, \quad x_*(t) \equiv x^+(t) \quad \forall t \geq 0.$$

Proof. For any admissible pair $(\bar{u}(t), \bar{x}(t))$, we have:

$$\Delta J = \int_0^{+\infty} e^{-\rho t} \left(F(\bar{x}(t)) - F(x^+(t))\right) dt \geq 0$$

as in this case $\bar{x}(t) \leq x^+(t) \leq a$ and $F(x)$ decreases for $x < a$. Theorem 2 is proved.

3 Investigation of the problem in case $x_0 > a > 1$

Results of the optimal control theory will be used in this section. Henceforth (except in section 4.2) we additionally assume that $F(x)$ is differentiable and $F'(x) < 0$ for $x < a$, $F'(a) = 0$, $F'(x) > 0$ for $x > a$. Also we suppose that the integral $\int_0^{+\infty} e^{-\rho t} F'(x(t))\, dt$ is convergent.

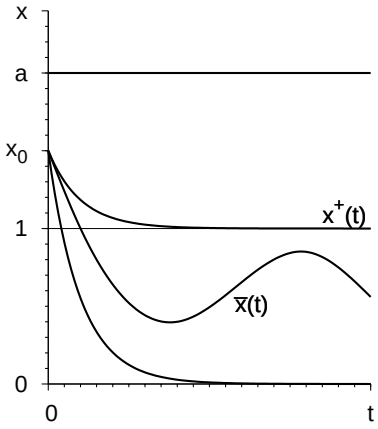


Figure 4: $x_*(t) = x^+(t)$ for $a > 1,\ 0 < x_0 \le a$.

Lemma 2 *In case $x_0 > a > 1$ for any admissible pair $(u(t), x(t))$ of problem (4) the trajectory $x(t)$ intersects the line $x = a$ once.*

Proof. First of all we note that the trajectory $x^+(t)$ is decreasing and tends to 1 when $t \to \infty$, so $x^+(t)$ intersects the line $x = a$ once in the point $\gamma = \ln\left(\dfrac{x_0 - 1}{a - 1}\right)$. We have two alternatives:

1. The trajectory $x(t) > 1$ for all $t \in [0, +\infty)$. Then

$$\dot{x}(t) = -x(t) + u(t) < 0 \quad \forall t \in [0, +\infty)$$

i.e., the function $x(t)$ is decreasing for $t \in [0, +\infty)$. We know that $x(t) \le x^+(t)$, whence $x(\gamma) \le x^+(\gamma) = a$. Therefore the trajectory $x(t)$ intersects the line $x = a$ once in the point $\theta \in [0, \gamma]$.

2. There exists a time moment σ such that $x(\sigma) = 1$ and $x(t) > 1$ for $t \in [0, \sigma)$. Similarly to 1^0 we can assert that $x(t)$ intersects the line $x = a$ once in the point $\theta \in [0, \sigma)$. The trajectory $x(t) \le 1$ for $t \in [\sigma, +\infty)$, since $u(t) \le 1$. This means that $x(t)$ does not intersect the line $x = a$ for $t \in [\sigma, +\infty)$. Lemma 2 is proved.

Lemma 3 *Let $(u(t), x(t))$ be an admissible pair of problem (4), $x(t_0) = x_0 > a > 1$ and θ be the moment of time where trajectory $x(t)$ intersects the line $x = a$, i.e., $x(\theta) = a$. If there exist two moments of time α_1, α_2: $\theta \le \alpha_1 < \alpha_2 < +\infty$ such that $u(t) < 1$, $t \in [\alpha_1, \alpha_2]$ then the control $u(t)$ is not optimal.*

Proof. It follows from lemma 2 that the moment of time θ exists. Denote

$$x(t) = \begin{cases} x_1(t), & 0 \le t < \theta \\ x_2(t), & \theta \le t < +\infty \end{cases}.$$

Here $x_1(\theta) = x_2(\theta) = a$. Let us take

$$\bar{u}(t) = \begin{cases} u(t), & 0 \le t < \theta \\ 1, & \theta \le t < +\infty \end{cases}$$

and the corresponding trajectory

$$\bar{x}(t) = \begin{cases} x_1(t), & 0 \le t < \theta \\ \bar{x}_2(t), & \theta \le t < +\infty \end{cases}$$

where $\bar{x}_2(t) = 1 + (a-1)e^{\theta - t}$. We have $x_2(t) \le \bar{x}_2(t)$. Then we obtain

$$\Delta J = J[\bar{u}] - J[u] = \int\limits_0^{+\infty} e^{-\rho t} F(\bar{x}(t))dt - \int\limits_0^{+\infty} e^{-\rho t} F(x(t))dt =$$

$$= \int\limits_\theta^{\alpha_1} e^{-\rho t} \left(F(\bar{x}_2(t)) - F(x_2(t))\right) dt + \int\limits_{\alpha_1}^{\alpha_2} e^{-\rho t} \left(F(\bar{x}_2(t)) - F(x_2(t))\right) dt + \int\limits_{\alpha_2}^{+\infty} e^{-\rho t} \left(F(\bar{x}_2(t)) - F(x_2(t))\right) dt.$$

$F(\bar{x}_2(t)) \le F(x_2(t))$ for $t \in [\theta, \alpha_1] \cup [\alpha_2, +\infty)$, due to $x_2(t) \le \bar{x}_2(t) \le a$ for such t and $F(\bar{x}_2(t)) < F(x_2(t))$ for $t \in (\alpha_1, \alpha_2)$, due to $x_2(t) < \bar{x}_2(t) < a$ for $t \in (\alpha_1, \alpha_2)$, because $F(x)$ is decreasing for $x < a$, whence $\Delta J < 0$. Therefore $u(t)$ is not optimal. Lemma 3 follows.

Corollary 1 *Let $(u(t), x(t))$ be an optimal pair of problem (4) and θ be the moment of time such that $x(\theta) = a$. Then $u(t) = 1$ for $t \in [\theta, +\infty)$.*

Further investigations are based on the Pontryagin's maximum principle.

3.1 Pontryagin's maximum principle

The Hamiltonian function for problem (4) has the following form:

$$K(t, x, \psi, u) = -e^{-\rho t} F(x) + \psi(-x + u).$$

Solving the optimization problem

$$K(t, x, \psi, u) \to \max_{u \in [0,1]}$$

we obtain

$$\bar{u}_*(\psi) = \begin{cases} 0, & \psi < 0 \\ 1, & \psi > 0 \\ [0,1], & \psi = 0 \end{cases}$$

i.e., the maximizer may be written as

$$\bar{u}_*(\psi) = h(\psi) \tag{5}$$

where $h(\cdot)$ is the Heaviside function:

$$h(s) = \begin{cases} 0, & s \le 0 \\ 1, & s > 0 \end{cases}.$$

Here $\psi = \psi(t)$ is the so-called conjugate variable that is a solution of the following conjugate equation:

$$\dot{\psi} = -K_x' = \psi + e^{-\rho t} F'(x). \tag{6}$$

Optimal control has the form (5).

Lemma 4 *Let $(u(t), x(t))$ be an admissible pair of problem (4), $x(t_0) = x_0 > a > 1$ and θ be the moment of time where trajectory $x(t)$ intersects the line $x = a$, i.e., $x(\theta) = a$. Then for arbitrary $\psi(0)$ the corresponding solution $\psi(t)$ of equation (6) can be equal to zero on the segment $[0, \theta]$ one time only.*

Proof. Let us suppose in contradiction that $\psi(t_1) = \psi(t_2) = 0$ for some $t_1, t_2 \in [0, \theta]$, $t_1 < t_2$. Using the Cauchy formula for equation (6) we find:

$$\psi(t) = e^t \left(\psi(0) + \int\limits_0^t e^{-s(1+\rho)} F'(x(s))ds \right).$$

Substituting t_1 and t_2 in the last formula we have:

$$\psi(t_1) = e^{t_1}\left(\psi(0) + \int\limits_0^{t_1} e^{-s(1+\rho)}F'(x(s))ds\right) = 0$$

$$\psi(t_2) = e^{t_2}\left(\psi(0) + \int\limits_0^{t_2} e^{-s(1+\rho)}F'(x(s))ds\right) = 0$$

whence

$$\int\limits_0^{t_1} e^{-s(1+\rho)}F'(x(s))ds = \int\limits_0^{t_2} e^{-s(1+\rho)}F'(x(s))ds.$$

It is clear that $t_1 = t_2$, since the integrand is negative. Lemma 4 is completely proved.

Corollary 2 *Optimal control of problem (4) is a piecewise constant function which can not have more than one point of switching.*

It follows from corollaries 1 and 2 that if the optimal control exists, then either $u_*(t) \equiv 1 \ \ \forall t \geq 0$ or

$$u_*(t) = \begin{cases} 0, & 0 \leq t < \tau_* \\ 1, & \tau_* \leq t < +\infty \end{cases}$$

where $\tau_* < \theta$ is the switching point. The trajectory that corresponds to this control has the following form:

$$x_*(t) = \begin{cases} x_0 e^{-t}, & 0 \leq t < \tau_* \\ 1 + (x_0 - e^{\tau_*})e^{-t}, & \tau_* \leq t < +\infty \end{cases}.$$

3.2 Boundary-value problem of maximum principle

Consider the following boundary-value problem of maximum principle:

$$\begin{cases} \dot{x} = K'_{\psi}|_{u=\bar{u}_*(\psi)} = -x + h(\psi), & x(0) = x_0 \\ \dot{\psi} = -K'_x|_{u=\bar{u}_*(\psi)} = \psi + e^{-\rho t}F'(x), & \psi(+\infty) = 0 \end{cases}. \tag{7}$$

The transversality condition $\psi(+\infty) = 0$ is, in general, a subject for discussion. In our case this particular form of transversality condition helps to solve the optimality problem. Other transversality conditions have been used in different articles; see for example [1], [2], [3]. Let the pair $(x(t), \psi(t))$ be the solution for problem (7). Then the pair $x(t), u(t) = \bar{u}_*(\psi)|_{\psi=\psi(t)} = h(\psi(t))$ is the optimal solution candidate for problem (4). Pontryagin's maximum principle is the necessary though not sufficient condition for optimality. Therefore a solution constructed via this maximum principle solution needs to be verified.

3.3 Finding the switching point

Suppose that $(x(t), \psi(t))$ is the solution for problem (7) and $x(t)$ has the form:

$$x(t) = \begin{cases} x_0 e^{-t}, & 0 \leq t < \tau_* \\ 1 + (x_0 - e^{\tau_*})e^{-t}, & \tau_* \leq t < +\infty \end{cases}.$$

This trajectory is defined by the following control:

$$u(t) = \begin{cases} 0, & 0 \leq t < \tau_* \\ 1, & \tau_* \leq t < +\infty \end{cases}.$$

In this section we discuss how to find the switching point τ_*. Solving the conjugate equation and using the transversality condition we obtain

$$\psi(t) = e^t \int\limits_{+\infty}^{t} e^{-s(1+\rho)} F'(x(s))ds.$$

The switching point τ_* is determined by the equation $\psi(\tau_*) = 0$. Thus

$$\int\limits_{+\infty}^{\tau_*} e^{-s(1+\rho)} F'(x(s))ds = 0.$$

Observe that $x(t) = 1 + (x_0 - e^{\tau_*})e^{-t}$ for $t \in [\tau_*, +\infty)$. Define $x_* = x(\tau_*)$ and make the following change of variables in the last integral:

$$x = x(s) = 1 + (x_0 - e^{\tau_*})e^{-s} = 1 + e^{\tau_* - s}(x_* - 1).$$

The last equality is valid as $x_0 e^{-\tau_*} = x_*$.

$$\int\limits_{+\infty}^{\tau_*} e^{-s(1+\rho)} F'(x(s))ds = -\frac{e^{-\tau_*(\rho+1)}}{(x_* - 1)^{\rho+1}} \int\limits_{1}^{x_*} F'(x)(x-1)^\rho dx.$$

Finally we arrive at

$$\int\limits_{1}^{x_*} F'(x)(x-1)^\rho dx = 0.$$

Taking into account the properties of the function $F'(x)$ we can assert that this equation has either a single solution for x_* or none. Let it have a solution. Then we can define $x_* > a$ such that feedback control is

$$u(x(t)) = \begin{cases} 0, & x(t) > x_* \\ 1, & x(t) \le x_* \end{cases}.$$

Using x_* we can find $\tau_* = \ln\left(\dfrac{x_0}{x_*}\right) > 0$, because $x_0 e^{-\tau_*} = x_*$.

4 The Dmitruk model in some special cases

This section describes two special cases of problem (4). The solution was found and the optimality was proved in both cases. The complete proof is complicated and omitted here. Note that theorems 1 and 2 remain valid; therefore we consider only the case $x_0 > a > 1$.

4.1 Special case 1

Suppose that $F(x) = (x-a)^4$ and $\rho = 1$. The function $F(x)$ is differentiable. So problem (4) takes the following form:

$$\begin{cases} \dot{x} = -x + u, \quad 0 \le t < +\infty \\ x(0) = x_0 \\ J = \int\limits_{0}^{+\infty} e^{-t}\ (x-a)^4 dt\ \to \min\limits_{u(\cdot)} \\ 0 \le u(t) \le 1 \end{cases}. \tag{8}$$

Let

$$d = \frac{(a-1)(135 + 60\sqrt{6})^{\frac{1}{3}}}{12} - \frac{5(a-1)}{4(135 + 60\sqrt{6})^{\frac{1}{3}}} - \frac{1}{4} + \frac{5}{4}a.$$

Theorem 3

1. *For* $1 < a < x_0 \le d$ *the following pair:*

$$x(t) = x^+(t) = (x_0 - 1)e^{-t} + 1$$
$$\psi(t) = -\frac{4}{5}(x_0 - 1)^3 e^{-4t} - 3(x_0 - 1)^2(1 - a)e^{-3t} - 4(x_0 - 1)(1 - a)^2 e^{-2t} - 2(1 - a)^3 e^{-t}$$
$$t \in [0, +\infty)$$

is the solution for problem (7).

2. *For* $x_0 > d$, $a > 1$ *the pair:*

$$x(t) = \begin{cases} x_0 e^{-t}, & 0 \le t < \tau_* \\ (x_0 - e^{\tau_*})e^{-t} + 1, & \tau_* \le t < +\infty \end{cases}$$

$$\psi(t) = \begin{cases} -\frac{4}{5}(x_0)^3 e^{-4t} + 3(x_0)^2 a e^{-3t} - 4x_0 a^2 e^{-2t} + 2a^3 e^{-t} + \\ +e^t \left[\frac{4}{5}(x_0)^3 e^{-5\tau_*} - 3(x_0)^2 a e^{-4\tau_*} + 4x_0 a^2 e^{-3\tau_*} - 2a^3 e^{-2\tau_*}\right], & 0 \le t < \tau_* \\ -\frac{4}{5}(x_0 - e^{\tau_*})^3 e^{-4t} - 3(x_0 - e^{\tau_*})^2(1 - a)e^{-3t} - \\ -4(x_0 - e^{\tau_*})(1 - a)^2 e^{-2t} - 2(1 - a)^3 e^{-t}, & \tau_* \le t < +\infty \end{cases}$$

is the solution for problem (7), where the switching point is given by:

$$\tau_* = \ln\left(\frac{x_0}{d}\right) > 0.$$

Note The optimality of the pair $(u(t) = h(\psi(t)), x(t))$, where $x(t)$ and $\psi(t)$ are functions defined by theorem 3, can be proved by means of the sufficient optimality conditions theorem (see [5]). Thus the following corollary ensues.

Corollary 3

1. *In case* $1 < a < x_0 \le d$ *optimal solution for problem (8) has the following form:*

$$u_*(t) \equiv 1, \quad x_*(t) = x^+(t) \quad \forall t \ge 0.$$

2. *In case* $x_0 > d$, $a > 1$ *optimal solution for the problem has the form:*

$$u_*(t) = \begin{cases} 0, & 0 \le t < \tau_* \\ 1, & \tau_* \le t < +\infty \end{cases}$$

$$x_*(t) = \begin{cases} x_0 e^{-t}, & 0 \le t < \tau_* \\ (x_0 - e^{\tau_*})e^{-t} + 1, & \tau_* \le t < +\infty \end{cases}$$

where

$$\tau_* = \ln\left(\frac{x_0}{d}\right) > 0.$$

4.2 Special case 2

Suppose that $F(x) = |x - a|$, so model (4) may be written in the form:

$$\begin{cases} \dot{x} = -x + u, \quad 0 \le t < +\infty \\ x(0) = x_0 \\ J = \int\limits_0^{+\infty} e^{-\rho t} \, |x - a| dt \; \to \min\limits_{u(\cdot)} \\ 0 \le u(t) \le 1 \end{cases} . \tag{9}$$

This problem is particularly interesting, because $F(x) = |x - a|$ is not differentiable in the point $x = a$. That means that direct application of the classical maximum principle to problem (9) is impossible. Let

$$D = \sqrt[\rho+1]{2}(a - 1) + 1.$$

Theorem 4

1. *For $1 < a < x_0 \le D$ the following pair:*

$$x(t) = x^+(t) = (x_0 - 1)e^{-t} + 1 \quad \forall t \ge 0$$

$$\psi(t) = \begin{cases} -\dfrac{1}{1+\rho} \, e^{-\rho t} + \dfrac{2}{1+\rho} \, e^{t - \theta(\rho+1)}, & 0 \le t < \theta \\ \dfrac{1}{1+\rho} \, e^{-\rho t}, & \theta \le t < +\infty \end{cases}$$

is the solution for problem (7), where

$$\theta = \ln\left(\frac{x_0 - 1}{a - 1}\right) > 0.$$

2. *For $x_0 > D$, $a > 1$ the pair:*

$$x(t) = \begin{cases} x_0 e^{-t}, & 0 \le t < \tau_* \\ 1 + (x_0 - e^{\tau_*})e^{-t}, & \tau_* \le t < +\infty \end{cases}$$

$$\psi(t) = \begin{cases} -\dfrac{1}{1+\rho} \, e^{-\rho t} + \dfrac{2}{1+\rho} \, e^{t - \theta(\rho+1)}, & 0 \le t < \theta \\ \dfrac{1}{1+\rho} \, e^{-\rho t}, & \theta \le t < +\infty \end{cases}$$

is the solution for problem (7), where

$$\tau_* = \ln\left(\frac{x_0}{D}\right) > 0$$

$$\theta = \ln\left(\frac{x_0 - e^{\tau_*}}{a - 1}\right) > 0.$$

Note The optimality of the pair $(u(t) = h(\psi(t)), x(t))$, where $x(t)$ and $\psi(t)$ are functions defined by theorem 4, can be established using the similar technique that was used for the proof of the sufficient optimality conditions theorem (see [5]).

Corollary 4

1. *In case* $1 < a < x_0 \leq D$ *optimal solution for problem (9) has the following form:*

$$u_*(t) \equiv 1, \quad x_*(t) = x^+(t) \quad \forall t \geq 0.$$

2. *In case* $x_0 > D$, $a > 1$ *optimal solution for the problem has the form:*

$$u_*(t) = \begin{cases} 0, & 0 \leq t < \tau_* \\ 1, & \tau_* \leq t < +\infty \end{cases}$$

$$x_*(t) = \begin{cases} x_0 e^{-t}, & 0 \leq t < \tau_* \\ 1 + (x_0 - e^{\tau_*})e^{-t}, & \tau_* \leq t < +\infty \end{cases}$$

where

$$\tau_* = \ln\left(\frac{x_0}{D}\right) > 0.$$

Acknowledgements. The author would like to extend her thanks to Assoc. Professor M.V. Orlov for very useful discussions and comments and to Professor A.V. Dmitruk for the problem statement. The linguistic assistance of Hugo A. van den Berg is gratefully acknowledged. The author would also like to express her gratitude to BHP Billiton company and to Dr Edwin H. van Leeuwen personally for the Scholarship.

References

1. Aseev, S.M., Kryazhimskiy, A.V.: The Pontryagin Maximum Principle and Optimal Economic Growth Problems. Proceedings of Steklov Math. Ins. of RAS , v. 257 pp. 1-272 (2007). In Russian

2. Aseev, S.M., Kryazhimskiy, A.V.: The Pontryagin Maximum Principle and Transversality Conditions for a Class of Optimal Control Problems With Infinite Time Horizons. SIAM J. Control and Optim., v. 43 pp. 1094-1119 (2004)

3. Aseev, S.M., Kryazhimskiy, A.V., Tarasyev, A.M.: The Pontryagin Maximum Principle and Transversality Conditions for a Class of Optimal Economic Growth Problems. Nonlinear Control Systems 2001: Proc. 5th IFAC Symp., St.-Petersburg, July 4-6, 2001. New York; London: Elsevier, v. 1 pp. 71-76 (2001)

4. Ashmanov, S.A.: Mathematical Models and Methods in Economics. Mosk. Gos. Univ., Moscow (1980). In Russian

5. Kiselev, Yu.N.: Sufficient Optimality Conditions Based on Pontryagin's Maximum Principle. Materials of Scientific Conference "Mathematical Models in Economics and Biology". Moscow: MAKS Press, pp. 57-67 (2003). In Russian

6. Kiselev, Yu.N., Orlov, M.V.: Investigation of One-Dimensional Problem of Resource Distribution on an Infinite Time Interval. Vestn. Mosk. Univ., Ser. 15: Comput. Math. Cybern., №. 2, pp. 43-48 (2007)

7. Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., Mishchenko, E.F.: The Mathematical Theory of Optimal Processes. Interscience, New York (1962)

УДК 517.444:519.2

О ПРОБЛЕМЕ ВОССТАНОВЛЕНИЯ КОЭФФИЦИЕНТА ПРЕЛОМЛЕНИЯ В ЗАДАЧАХ ДИФРАКЦИОННОЙ ТОМОГРАФИИ

© 2009 г. О. В. Шестаков

oshestakov@cs.msu.su

Кафедра Математической Статистики

1 Введение

В задачах вычислительной томографии возникает проблема восстановления изображения внутренней структуры объекта по проекционным данным (см. [1]). Если для получения данных используются рентгеновские или γ-лучи, то проблема сводится к обращению преобразования Радона. Однако в некоторых приложениях вред, который наносит излучение, может превзойти ту пользу, которую дает полученная информация. В таких случаях применяется ультразвук или электромагнитные волны, считающиеся относительно безопасными. При этом уже нельзя игнорировать волновую природу излучения.

Дифракционная томография представляет собой обобщение обычной вычислительной томографии. Вместо пучка прямых лучей в томографическом эксперименте используется дифракционное волновое поле, и требуется восстановить коэффициент преломления объекта. Дифракционная томография основана на линейном интегральном преобразовании (дифракционном преобразовании), которое играет ту же роль, что и преобразование Радона в обычной томографии.

Пусть $n(x) = (1+f(x))^{1/2}$ $(x \in \mathbf{R}^2)$ — коэффициент преломления объекта. Зарегистрировав рассеянную волну за пределами объекта, требуется восстановить функцию $f(x)$. Известно (см. ниже), что по полученным данным можно восстановить (в рамках линейной аппроксимации) преобразование Фурье от $f(x)$ внутри некоторого круга. Естественно, для восстановления $f(x)$ требуется знать ее преобразование Фурье всюду. В общем случае для этого требуется использовать аналитическое продолжение, что представляет собой очень неустойчивую задачу.

Мы рассматриваем специальный случай, когда $f(x)$ представляет собой вероятностную плотность на компактном носителе. В работе [2] показано, что в этом случае удается избежать значительного повышения неустойчивости. В той же работе предложен метод восстановления функции $f(x)$.

В данной работе мы модифицируем метод, предложенный в [2], и получим оценку скорости его сходимости.

2 Постановка задачи дифракционной томографии

Пусть $f(x) \geq 0$ и имеет носителем круг единичного радиуса с центром в начале координат. Будем предполагать, что $f(x)$ представляет собой вероятностную плотность. Объект облучается волной $e^{-ikt}u_I(x)$ с частотой k. Мы рассматриваем только плоские волны $u_I(x) = e^{ik(\theta,x)}$, где $\theta \in S^1$ — направление распространения волны.

Для данной функции $f(x)$ можно получить рассеянную волну $e^{-ikt}u_s(x)$, для которой $u = u_I + u_s$ удовлетворяет волновому уравнению

$$\triangle u + k^2(1+f)u = 0, \tag{1}$$

где $\triangle$ — это оператор Лапласа. Чтобы решить уравнение (1) в рамках аппроксимации Рытова (см. [1]), положим $u = u_I e^{k\omega}$. Имеем

$$k\triangle\omega + 2ik^2(\theta, \nabla\omega) + k^2\left|\nabla\omega\right|^2 = -k^2 f,$$

где ∇ обозначает градиент. Отбрасывая член с $|\nabla\omega|^2$, получаем аппроксимацию Рытова в форме $u_R = u_I e^{k\omega_R}$, где ω_R удовлетворяет уравнению

$$\triangle(u_I\omega_R) + k^2(u_I\omega_R) = -kfu_I. \tag{2}$$

Решение этого дифференциального уравнения может быть найдено с помощью функции Грина, которая в данном случае равна

$$H_0(k\,|x|) = -\frac{i}{4\pi}\int\limits_{\mathbf{R}^1} e^{i(|x_1|a(\rho)+x_2\rho)}\frac{d\rho}{a(\rho)}. \tag{3}$$

Здесь $4iH_0$ — функция Ханкеля первого рода нулевого порядка, а $a(\rho) = \sqrt{k^2-\rho^2}$. Решение уравнения (2) принимает следующий вид

$$u_I(x)w_R(x) = -k\int\limits_{\mathbf{R}^2} H_0(k\,|x-y|)f(y)u_I(y)dy. \tag{4}$$

В задаче дифракционной томографии требуется найти f, зная u_s за пределами объекта. При использовании аппроксимации Рытова для этого требуется решить уравнение (4) относительно f при заданной функции $g(\theta,s) = w_R(r\theta + s\theta^\perp)$, где $r > 1$ — фиксированное число (т. е. ω_R измеряется за пределами диска радиуса r). Подставляя (3) в (4) и используя то, что $u_I(x) = e^{ik(\theta,x)}$, имеем

$$g(\theta,s) = -ke^{-ikr}\int\limits_{\mathbf{R}^2} H_0\left(k(\sqrt{(r-r')^2+(s-s')^2}\right)f(r'\theta+s'\theta^\perp)e^{ikr'}dr'ds' =$$

$$= \frac{ik}{4\pi}e^{-ikr}\int\limits_{\mathbf{R}^3} e^{i(|r-r'|a(\rho)+(s-s')\rho)}\frac{d\rho}{a(\rho)}f(r'\theta+s'\theta^\perp)e^{ikr'}dr'ds'.$$

Поскольку $r > 1$ и $f(r'\theta + s'\theta^\perp) = 0$ для $r' > 1$, можно опустить знак модуля в выражении $|r-r'|$. Меняя порядок интегрирования, получаем

$$g(\theta,s) = \frac{ik}{4\pi}e^{-ikr}\int\limits_{\mathbf{R}} e^{is\rho}\frac{e^{ira(\rho)}}{a(\rho)}\int\limits_{\mathbf{R}^2} e^{-i(r'(a(\rho)-k)+s'\rho)}f(r'\theta+s'\theta^\perp)dr'ds'd\rho.$$

Это и есть дифракционное преобразование. Интегрирование по $y = r'\theta + s'\theta^\perp$ представляет собой преобразование Фурье в $\mathbf{R}^2$. Следовательно,

$$g(\theta,s) = \frac{ik}{4\pi}e^{-ikr}\int\limits_{\mathbf{R}} e^{is\rho}\frac{e^{ira(\rho)}}{a(\rho)}\hat{f}((a(\rho)-k)\theta+\rho\theta^\perp)d\rho.$$

Интегрирование по ρ представляет собой обратное преобразование Фурье в $\mathbf{R}$. Следовательно,

$$\hat{f}((a(\rho)-k)\theta+\rho\theta^\perp) = -2i\frac{a(\rho)}{k}e^{ir(k-a(\rho))}\hat{g}(\theta,\rho), \tag{5}$$

где $\hat{g}(\theta,\rho)$ — одномерное преобразование Фурье функции $g(\theta,s)$ по второй переменной. При изменении ρ в интервале $[-k,k]$ значение $(a(\rho)-k)\theta+\rho\theta^\perp = \sqrt{k^2-\rho^2}\theta+\rho\theta^\perp - k\theta$ меняется на полуокружности с центром $-k\theta$, которая делится началом координат на равные части. Соответственно, если θ принимает значения из S^1, то выражение (5) определяет $\hat{f}$ в круге радиуса не меньше k. Если $k\to\infty$, то $\hat{f}$ определена полностью, и мы приходим к задаче обычной вычислительной томографии (см. [1]).

3 Метод реконструкции

Поскольку $f(x)$ представляет собой вероятностную плотность, $\hat{f}(-\tau)$ ($\tau = (t_1, t_2)$) представляет собой ее характеристическую функцию. Из предыдущего раздела следует, что $\hat{f}(\tau)$ известна в круге радиуса k: $|\tau| < k$.

Пусть $N = 2n$, где n — натуральное число, и $\theta_1, \dots \theta_N \in S^1$ — направления на плоскости $\mathbf{R}^2$, выбранные следующим образом:

$$\theta_j = (\nu_j, -1)/(\nu_j^2 + 1)^{1/2}, \qquad j = 1, \dots n,$$

$$\theta_j = (1, \nu_{j-n})/(\nu_{j-n}^2 + 1)^{1/2}, \qquad j = n+1, \dots 2n,$$

где

$$\nu_k = \cos(\pi(2k-1)/(2n)), \qquad k = 1, \dots, n,$$

интерполяционные узлы Чебышева. Пусть X — случайная величина с плотностью $f(x)$. Функция $\varphi_j(\rho) = \hat{f}(\rho\theta_j)$ ($\rho \in \mathbf{R}$, $j = 1, ..., N$) является характеристической функцией случайной величины $X_{\theta_j} = -(\theta_j, X)$. Так как $\hat{f}(\tau)$ известна в круге радиуса k, $\varphi_j(\rho)$ известна на отрезке $[-k, k]$. Поскольку плотность $f(x)$ имеет компактный носитель, функция $\varphi_j(\rho)$ бесконечно дифференцируема. Разложим $\varphi_j(\rho)$ в ряд Тейлора в точках k и $-k$ до члена порядка $m-1$:

$$\varphi_j(\rho) = P_{j,m}^{\pm}(\rho) + \frac{\varphi_j^{(m)}(\xi^{\pm})(\rho \mp k)^m}{m!}, \quad |\rho| > k,$$

где $\xi^+ \in (k, \rho)$, $\xi^- \in (\rho, -k)$ и

$$P_{j,m}^{\pm}(\rho) = \sum_{l=0}^{m-1} \varphi_j^{(l)}(k) \frac{(\rho \mp k)^l}{l!}.$$

Определим функции $\psi_{j,m}(\rho)$:

$$\psi_{j,m}(\rho) = \begin{cases} \varphi_j(\rho) \text{ при } \rho \in [-k, k], \\ P_{j,m}^{+}(\rho) \text{ при } \rho > k, \\ P_{j,m}^{-}(\rho) \text{ при } \rho < k. \end{cases}$$

Обозначим через $\phi_\sigma(x)$ двумерную плотность нормального распределения:

$$\phi_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x_1^2 + x_2^2}{2\sigma^2}\right), \; x = (x_1, x_2) \in \mathbf{R}^2.$$

Если плотность $f(x)$ достаточно гладкая, то $f * \phi_\sigma(x)$ хорошо приближает $f(x)$ при малых значениях σ.

Пусть $\psi_{n,m}(\theta, \rho) = \psi_{j,m}(\rho)$ для всех $\rho \in \mathbf{R}$ и всех θ, принадлежащих пучку направлений от $\pm\theta_j - \Delta\theta$ до $\pm\theta_j + \Delta\theta$, $j = 1, ..., N$, где $\pm\theta_j - \Delta\theta$ и $\pm\theta_j + \Delta\theta$ — направления, отстоящие на угол $\pi/(2n)$ влево и вправо от направления $\pm\theta_j$. Запишем $\psi_{n,m}(\theta, \rho)$ в декартовых координатах: $\psi_{n,m}(t_1, t_2)$, и в качестве аппроксимации для $f * \phi_\sigma(x)$ будем использовать модуль обратного преобразования Фурье от произведения $\psi_{n,m}(t_1, t_2)\hat{\phi}_\sigma(t_1, t_2)$, который обозначим через $|f_{n,m,\sigma}(x)|$. Ниже мы приведем оценку близости между $|f_{n,m,\sigma}(x)|$ и $f * \phi_\sigma(x)$.

4 Оценка скорости сходимости метода реконструкции

Теорема. Пусть $f(x)$ — плотность двумерного вероятностного распределения с носителем в единичном круге с центром в начале координат, и пусть функция $|f_{n,m,\sigma}(x)|$ получена по методу, описанному в предыдущем разделе. Тогда

$$\sup_{x\in\mathbf{R}^2}|f*\phi_\sigma(x)-|f_{n,m,\sigma}(x)||\le\frac{\sqrt{\pi}}{2\sqrt{2}\sigma^3 n}+$$

$$+\frac{e^{-\sigma^2k^2}}{2\sqrt{\pi}}\left(\frac{1}{2^{m/2}\sigma^{m+2}\Gamma\left(\frac{m+1}{2}\right)}+\frac{k}{2^{(m+1)/2}\sigma^{m+1}\Gamma\left(\frac{m+2}{2}\right)}\right), \tag{6}$$

где $\Gamma(z)$ — гамма-функция Эйлера.

Если дополнительно предположить, что $f(x)$ непрерывно дифференцируема, тогда

$$\sup_{x\in\mathbf{R}^2}|f(x)-|f_{n,m,\sigma}(x)||\le C_f(\pi/2)^{1/2}\sigma+\frac{\sqrt{\pi}}{2\sqrt{2}\sigma^3 n}+$$

$$+\frac{e^{-\sigma^2k^2}}{2\sqrt{\pi}}\left(\frac{1}{2^{m/2}\sigma^{m+2}\Gamma\left(\frac{m+1}{2}\right)}+\frac{k}{2^{(m+1)/2}\sigma^{m+1}\Gamma\left(\frac{m+2}{2}\right)}\right), \tag{7}$$

где $C_f=\sup\limits_{x\in\mathbf{R}^2}|\nabla f(x)|$.

Доказательство. Во-первых заметим, что если $f(x)$ непрерывно дифференцируема, то (см. [3])

$$\sup_{x\in\mathbf{R}^2}|f(x)-f*\phi_\sigma(x)|\le C_f(\pi/2)^{1/2}\sigma.$$

Поэтому (7) следует из (6).

Докажем (6). Поскольку $\varphi_j(\rho)$ является характеристической функцией случайной величины с носителем в $[-1,1]$, она бесконечно дифференцируема и $\left|\varphi_j^{(m)}(\rho)\right|\le 1$ для всех $\rho\in\mathbf{R}$ и всех натуральных m.

Далее при произвольном $\theta\in S^1$

$$\left|\hat f(\rho\theta)-\psi_{n,m}(\theta,\rho)\right|\le\left|\hat f(\rho\theta)-\varphi_j(\rho)\right|+|\varphi_j(\rho)-\psi_{j,m}(\rho)|\le$$

$$\begin{cases}\frac{\pi|\rho|}{n},\ \text{при}\ |\rho|\le k,\\ \frac{\pi|\rho|}{n}+\frac{(|\rho|-k)^m}{m!},\ \text{при}\ |\rho|>k.\end{cases}$$

Следовательно, для любого $x\in\mathbf{R}^2$

$$|f*\phi_\sigma(x)-|f_{n,m,\sigma}(x)||\le|f*\phi_\sigma(x)-f_{n,m,\sigma}(x)|=$$

$$=(2\pi)^{-2}\left|\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}e^{-i(t_1x_1+t_2x_2)}(\hat f(t_1,t_2)-\psi_{n,m}(t_1,t_2))e^{-\sigma^2(t_1^2+t_2^2)/2}dt_1dt_2\right|\le$$

$$\le(2\pi)^{-2}\int_{S^1}d\theta\int_0^{\infty}\left|\hat f(\rho\theta)-\psi_{n,m}(\theta,\rho)\right|e^{-\sigma^2\rho^2/2}\rho d\rho\le$$

$$\le(2\pi)^{-1}\left(\int_0^k\frac{\pi}{n}e^{-\sigma^2\rho^2/2}\rho^2d\rho+\int_k^{\infty}\left(\frac{\pi}{n}\rho+\frac{(\rho-k)^m}{m!}\right)e^{-\sigma^2\rho^2/2}\rho d\rho\right)=$$

$$= (2\pi)^{-1}\left(\int\limits_0^\infty \frac{\pi}{n} e^{-\sigma^2\rho^2/2}\rho^2 d\rho + \int\limits_0^\infty \frac{\rho^m}{m!} e^{-\sigma^2(\rho+k)^2/2}(\rho+k)d\rho\right) \leq$$

$$\leq \frac{\sqrt{\pi}}{2\sqrt{2}\sigma^3 n} + (2\pi)^{-1}\int\limits_0^\infty \frac{\rho^m}{m!} e^{-\sigma^2(\rho^2+k^2)/2}(\rho+k)d\rho =$$

$$= \frac{\sqrt{\pi}}{2\sqrt{2}\sigma^3 n} + \frac{e^{-\sigma^2k^2/2}}{2\pi m!}\left(\frac{2^{m/2}\Gamma\left(\frac{m+2}{2}\right)}{\sigma^{m+2}} + \frac{k2^{(m-1)/2}\Gamma\left(\frac{m+1}{2}\right)}{\sigma^{m+1}}\right). \quad (8)$$

Подставляя в (8) выражение

$$m! = \Gamma(m+1) = \frac{2^m}{\sqrt{\pi}}\Gamma\left(\frac{m+1}{2}\right)\Gamma\left(\frac{m+2}{2}\right),$$

получаем (6). Теорема доказана.

Как отмечалось в [2] и [3], использование конечного числа направлений не позволяет однозначно восстановить функцию $f(x)$. Однако в случае, когда $f(x)$ представляет собой вероятностную плотность, неравенство (6) позволяет оценить степень неопределенности в реконструкции. Если к тому же известно, что $f(x)$ имеет необходимую гладкость, то можно использовать оценку (7). Выбрав достаточно большие m и n и маленькое σ, точность реконструкции можно сделать сколь угодно большой.

Список литературы

[1] A. C. Kak, M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[2] L. B. Klebanov, S. T. Rachev. *On a special case of the basic problem in diffraction tomography*//Stochastic Models, 1996, Vol. 12, N. 2, P. 181-197.

[3] L. A. Khalfin, L. B. Klebanov. *A solution of the computer tomography paradox and estimating the distances between the densities of measures with the same marginals*//The Annals of Probability, 1994, Vol. 22, N. 4, P. 2235-2241.

УДК 004.67:575

IrGene 1.0 — ИНТЕРФЕЙС И ПРОГРАММА ДЛЯ АНАЛИЗА ПОПУЛЯЦИОННО-ГЕНЕТИЧЕСКИХ ДАННЫХ В ИММУНОЛОГИИ

© 2009 г. Е. А. Сытин

esytin@cs.msu.su

Кафедра Вычислительных технологий и моделирования

1 Введение

В коротком плече 6-й хромосомы человека локализован главный комплекс тканевой совместимости — HLA (от англ. *human leukocyte antigens*) — генетический локус, содержащий более 4 тыс пар оснований. Известно, что HLA является центральным генетическим аппаратом для функционирования иммунной системы и, кроме того, различия по данному участку генома обуславливают наиболее резкую несовместимость тканей при пересадках (отсюда и название локуса). В системе HLA выделяют две основные области — HLA-I и HLA-II [5, 9].

На уровне популяции HLA обладает выраженным полиморфизмом, благодаря чему обеспечивается антигенная индивидуальность организма. HLA содержит около 20 аллельных генов. Аллельное разнообразие главного комплекса тканевой совместимости постоянно уточняется. По состоянию на 2009 г., выделено более 2700 различных аллелей генов HLA, т. е. в среднем 135 аллелей на один аллельный ген [10]. Считается, что вероятность HLA-идентичности для двух произвольно выбранных людей составляет не более 10^{-6} [8].

Продукты главного комплекса тканевой совместимости, имеющие признаки антигенной специфичности (т. е. HLA-антигены), образуют HLA-фенотип. Ранние исследования системы HLA базировались, главным образом, на изучении полиморфизмов HLA-I с использованием серологического (т. е. клеточно-опосредованного) типирования. Открытие в середине 1980-х полимеразной цепной реакции позволило перейти от характеристики структуры HLA по экспрессируемым продуктам генов к непосредственному анализу строения локуса методами ДНК-типирования.

Выявлены ассоциации инфекционных, аутоиммунных и других заболеваний с наличием того или иного HLA-антигена [1], что, в частности, обусловлено участием HLA-антигенов в иммунном ответе в качестве корецепторов двойного распознавания (Т-рестрикция). Указанные ассоциации не являются детерминированными и носят статистический характер. Согласно одной из гипотез, формирование разнообразия HLA-фенотипов в процессе эволюции явилось механизмом повышения видовой устойчивости против инфекционных агентов [4].

О наличии или отсутствии наследственной предрасположенности к заболеваниям судят различными способами — например, по величине относительного риска [12]. Относительный риск развития заболеваний у носителей некоторых аллелей генов HLA может возрастать в 1,7–90 раз [5]. Частотные распределения аллелей и соответствующие им величины относительного риска могут варьировать на межэтническом и внутриэтническом уровнях, а также в зависимости от географической локализации исследуемой группы [8].

Соматические клетки человека являются диплоидными, т. е. содержат два экземпляра каждого гена — по одному на “материнской” и “отцовской” хромосомах. Наборы генов HLA из каждой хромосомы кодоминантны, т. е. одинаково влияют на HLA-фенотип. Совпадение аллельных вариантов некоторого гена принято называть гомозиготностью по данному гену. Наличие гомозиготности по генам HLA может быть ассоциировано с повышенной восприимчивостью к заболеваниям [4]. В современных работах по иммуногенетике рассматривается понятие функциональной гомозиготности и установлены связи наследственной предрасположенности к заболеваниям с присутствием в генотипе т. н. маркерных аллелей — сочетаний определённых (возможно, различных) аллелей одного гена [1].

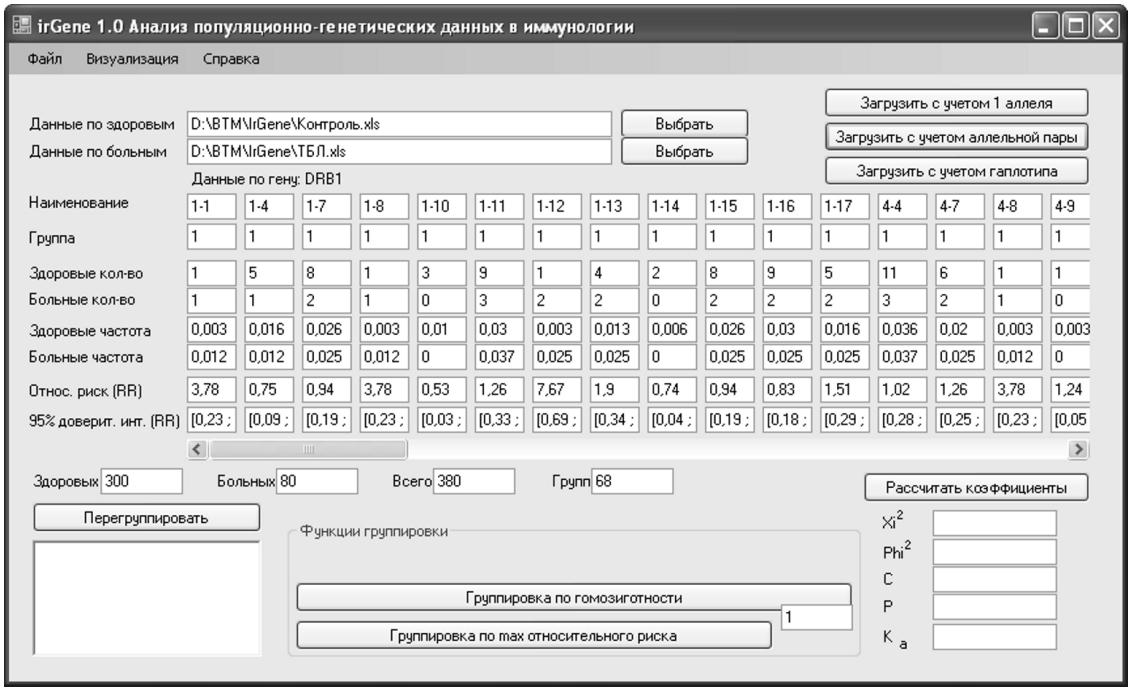


Рис. 1: Внешний вид программы-оболочки IrGene 1.0

Ген 1		...	Ген k		Кол-во
Аллель 1	Аллель 2		Аллель 1	Аллель 2	
1	1	...	1	1	$n_{11...11}$
1	1	...	1	2	$n_{11...12}$
...	...	...	...	...	...
m_1	m_1	...	m_k	m_k	$n_{m_1m_1...m_km_k}$

Таблица 1: Структура данных

Для удобства анализа популяционного полиморфизма системы HLA и попарных межгрупповых сравнений частотных распределений генотипических признаков предлагается программа-оболочка IrGene 1.0 (от англ. *immune response gene* — ген иммунного ответа). Ниже приводится описание возможностей и интерфейса программы с результатами её применения для решения одной прикладной задачи.

2 Описание программы

Программа IrGene 1.0 написана на языке *C#* в среде разработки Microsoft Visual Studio 2005 и доступна в интернет по адресу http://immunol.inm.ras.ru/old/IrGene. Для её нормального функционирования под Windows XP необходимо установить Framework.net 2 и MS Excel 2007. Исполняемый модуль программы имеет название irgene.exe. Внешний вид программы-оболочки показан на рис. 1.

Выбор данных HLA-типирования для сравнительного анализа осуществляется при помощи кнопок “Выбрать” с явным указанием интересующих файлов данных, которые хранятся в директории пользователя в виде таблиц Excel (табл. 1). Исходно предполагалось, что программа будет использована для анализа различий полиморфизмов HLA в группах здоровых и больных людей. Поэтому данные условно обозначены как “Данные по здоровым” и “Данные по

	Специфичность 1	Специфичность 2	Всего
Здоровые	a	b	$a+b$
Больные	c	d	$c+d$
Итого	$a+c$	$b+d$	$n=a+b+c+d$

Таблица 2: Четырёхпольная таблица сопряжённости

больным". Заголовок таблиц содержит названия генов системы HLA, доступных для анализа. В остальных строках перечислены всевозможные генотипы (относительно рассматриваемой группы генов) с указанием количества индивидов для каждого генотипа.

После указания данных имеется возможность загрузки таблицы сопряжённости признаков для аллелей, аллельных пар или гаплотипов — выбором одной из кнопок в правом верхнем углу окна пользователя (рис. 1). В появившемся списке необходимо выбрать названия одного или нескольких интересующих генов.

Таблица сопряжённости содержит две строки и не менее двух столбцов, формируется в терминах абсолютных значений и частот, и для неё рассчитываются параметры относительного риска. Результаты отображаются в окне пользователя, предусмотрена возможность экспорта в формат Excel (пункты меню "Файл" и "Сохранить результат"). Графическая визуализация данных возможна через пункты меню "Визуализация" и "Построить диаграмму". В случае если получаемая таблица сопряжённости шире размеров окна, то возможность её просмотра реализуется при помощи поля прокрутки. Ниже поля прокрутки в окне пользователя приводятся данные, характеризующие размеры выборки (поля "Здоровых", "Больных", "Всего") и количество столбцов в таблице сопряжённости (поле "Групп") (рис. 1).

Значимость различий между частотными распределениями признаков устанавливается в программе с использованием статистики χ^2 и величины относительного риска (OP). Величина OP, равная ad/bc, где a, b, c и d — элементы соответствующей четырёхпольной таблицы сопряжённости (табл. 2), показывает, во сколько раз чаще встречается заболевание при наличии данной специфичности, чем при её отсутствии [12]. Границы 95%-ного доверительного интервала (CI) для относительного риска рассчитываются по формуле 95%CI для $\ln(\mathrm{OP}) = \ln(\mathrm{OP}) \pm 1{,}96 \times (1/a + 1/b + 1/c + 1/d)^{1/2}$. Если одно из полей таблицы сопряжённости равно нулю, то для расчёта относительного риска используется поправка Холдейна: $\mathrm{OP} = (a+0{,}5)(d+0{,}5)/(b+0{,}5)(c+0{,}5)$.

Маркерные специфичности, ассоциированные со значимым увеличением или снижением относительного риска, отмечаются в окне пользователя "подкрашиванием" границ доверительного интервала красным или зелёным цветом соответственно.

В программе предусмотрена возможность автоматической группировки аллелей, аллельных пар и гаплотипов путем генерации таблицы сопряжённости 2×2 объединением произвольно выбранного количества специфичностей, имеющих наибольшую величину относительного риска. В случае анализа аллельных пар имеется возможность группировки по гомозиготности. Существует способ ручной группировки — явным указанием номера группы (в полях "Группа") для каждой специфичности и нажатием кнопки "Перегруппировать".

Кнопка "Рассчитать коэффициенты" в правом нижнем углу окна пользователя позволяет оценить величину χ^2 с поправкой Йейтса на непрерывность [13]:

$$\chi^2 = \sum \frac{(|R_{\text{набл}} - R_{\text{ож}}| - \frac{1}{2})^2}{R_{\text{ож}}},$$

где $R_{\text{набл}}$ и $R_{\text{ож}}$ — наблюдаемое и ожидаемое значения в графе таблицы сопряжённости, суммирование берётся по всем графам за исключением "Итого" и "Всего". Также определяются показатель взаимной сопряжённости $\varphi^2 = \chi^2/n$ — для исключения влияния количества наблюдений на величину связи между признаками [3], коэффициент взаимной сопряжённости Чупрова $C = \sqrt{\varphi^2/\sqrt{(k-1)(s-1)}}$ — для исключения влияния числа степеней свободы k и s, уровень значимости p для критерия χ^2 и коэффициент ассоциации Юла, характеризующий

Коэф-т ассоциации (k_a)	0,1–0,3	0,3–0,5	0,5–0,7	0,7–0,9	0,9–0,99
Качеств. характеристика силы связи	Слабая	Умеренная	Заметная	Высокая	Весьма высокая

Таблица 3: Шкала Чеддока — качественная характеристика силы связи между признаками

силу связи между признаками: $k_a = (ad - bc)/(ad + bc)$. Для качественной характеристики силы связи по величине коэффициента ассоциации применяется шкала Чеддока (табл. 3). Информационное поле в левом нижнем углу служит для уточнения содержимого других полей окна пользователя.

Если значение одного из полей таблицы меньше 5, то вместо критерия χ^2 в программе используется точный двусторонний критерий Фишера (на данный момент — только для четырёхпольных таблиц сопряжённости).

3 Некоторые результаты

В работе [6] с использованием программы IrGene 1.0 по данным ДНК-типирования низкого разрешения описаны HLA DRB1-маркеры наследственной предрасположенности к развитию туберкулёза лёгких (ТБЛ). Исследование образцов периферической крови, полученных от 300 здоровых доноров крови из г. Москвы [2] и от 80 больных туберкулёзом лёгких, находившихся на стационарном лечении в НИИ фтизиопульмонологии ММА имени И.М. Сеченова, проводилось в отделе иммуногенетики Института иммунологии ФМБА России. Геномную ДНК выделяли методом высаливания по стандартной процедуре [11]. HLA генотипирование образцов ДНК проводили методом мультипраймерной полимеразной цепной реакции [7]. Для типирования гена DRB1 главного комплекса тканевой совместимости класса II использовали наборы праймеров HLA-ДНК-Тех (НПФ “ДНК-Технология”, Россия).

Получено [6], что с повышенной восприимчивостью к туберкулёзу значимо ассоциированы четыре аллельные пары DRB1-специфичностей (04/15, 04/16, 11/17 и 13/15). Анализ четырёхпольной таблицы сопряжённости, в которой указанные генотипы были объединены в одну группу, показал наличие заметной силы связи между изучаемыми признаками по шкале Чеддока ($k_a = 0{,}63$). Аллельная группа DRB1*13 оказалась значимо ассоциирована с восприимчивостью (OP=1,57, 95%CI=[1,01; 2,46]), а DRB1*11 — с резистентностью к туберкулёзу (OP=0,49, 95%CI=[0,26; 0,92]). Связь восприимчивости с гомозиготностью по DRB1 отсутствовала. Сделан вывод, что аллельный ген DRB1, относящийся к HLA локусу класса II, активно участвует в патогенезе туберкулёзного процесса.

4 Обсуждение и выводы

В исследованиях последних лет выявлены генетические основы резистентности к заболеваниям и показана ведущая роль главного комплекса тканевой совместимости как основного генетического аппарата, определяющего функционирование иммунной системы человека [5, 9]. В настоящее время в России ведётся работа по формированию банка данных HLA-типирования, что в перспективе позволит составить подробное представление об этнотерриториальных особенностях генетического полиморфизма HLA. Программа IrGene 1.0 предназначена для описания и сравнительного анализа генетических полиморфизмов системы HLA на популяционном уровне. Она отличается простым интерфейсом и представляет собой удобный инструмент обработки и анализа данных в исследованиях по проблеме “HLA и болезни”.

5 Благодарности

Автор благодарит гл. науч. сотр. НИИ фтизиопульмонологии ММА имени И.М. Сеченова д.м.н., проф. Р.П. Селицкую и вед. науч. сотр. отдела иммуногенетики ГНЦ Институт им-

мунологии ФМБА России д.м.н. М.Н. Болдыреву за консультации и предоставленный набор данных для тестирования программы.

Список литературы

[1] Болдырева М.Н. HLA (класс II) и естественный отбор. "Функциональный" генотип, гипотеза преимущества "функциональной" гетерозиготности. Автореф. дисс. ... д-ра мед. наук. М., 2007. 47 с.

[2] Болдырева М.Н., Алексеев Л.П., Хаитов Р.М. и соавт. HLA-генетическое разнообразие населения России и СНГ. I. Русские // Иммунология. 2005. № 5. С. 260–267.

[3] Гублер Е.В., Генкин А.А. Применение непараметрических критериев статистики в медико-биологических исследованиях. Л.: Медицина, 1973. 144 с.

[4] Маянский Н.А., Маянский А.Н. Номенклатура и функции главного комплекса гистосовместимости человека // Иммунология. 2006. № 1. С. 43–46.

[5] Петров Р.В. Иммунология. М.: Медицина, 1987. 416 с.

[6] Селицкая Р.П., Болдырева М.Н. и соавт. Оценка корреляционных отношений полиморфизма DRB1-локуса системы HLA и восприимчивости к туберкулёзу // Иммунология. 2009 (представлено к публикации).

[7] Трофимов Д.Ю. Разработка метода мультипраймерной ПЦР для типирования генов HLA II класса: Дисс. ... канд. биол. наук. М., 1996.

[8] Хаитов Р.М., Алексеев Л.П. Предназначение иммунной системы: выполнение физиологических функций, обеспечивающих генетическое постоянство внутренней среды организма // Физиология и патология иммунной системы. 2004. № 8. С. 3–14.

[9] Ярилин А.А. Основы иммунологии. М.: Медицина, 1999. 608 с.

[10] Holdsworth R., Hurley C.K., Marsh S.G.E. et al. The HLA dictionary 2008: a summary of HLA-A, -B, -C, -DRB1/3/4/5, and -DQB1 alleles and their association with serologically defined HLA-A, -B, -C, -DR, and -DQ antigens // Tissue Antigens. 2009. V. 73, № 2. P. 95–170.

[11] Miller S.A., Dykes D., Polesky H.F. A simple salting-out procedure for extracting DNA from human nucleated cells // Nucleic Acids Res. 1988. V. 16. P. 1215.

[12] Woolf B. On estimating the relation between blood group and disease // Ann. Hum. Genet. 1955. V. 19. P. 251–253.

[13] Yates F. Contingency tables involving small numbers and the χ^2 test // Suppl. J. Roy. Statist. Soc. 1934. V. 1. P. 217.

УДК 004.457

ПРОГРАММИРУЕМЫЙ АГЕНТ ВЗАИМОДЕЙСТВИЯ ПО ПРОТОКОЛУ ПЕРЕДАЧИ ГИПЕРТЕКСТА

© 2009 г. Ю. В. Власенко, А. В. Столяров

yuliavlasenko@intelib.org, avst@cs.msu.ru

Кафедра алгоритмических языков

1 Мотивация и постановка задачи

Комплекс сетевых служб, основанных на протоколе передачи гипертекста (HTTP), возникший в начале девяностых годов прошедшего века, за полтора десятилетия своего существования непрерывно и весьма интенсивно развивался; в своём современном виде комплекс используется для решения самых разнообразных информационно-технических задач, в том числе и таких, для которых протокол HTTP исходно не был предназначен.

Изначально HTTP задуман как протокол передачи информации, организованной в виде гипертекста и хранящейся на HTTP-серверах в виде наборов файлов, клиенту, представляющему собой пользовательскую программу визуализации гипертекста (браузер). В современных реалиях очень часто информация на сервере генерируется динамически непосредственно перед отправкой клиенту. Сама эта информация также может содержать не только гипертекст, но и информацию, представленную в практически произвольном формате — изображения, архивы, аудио- и видеофайлы и т.п. вплоть до бинарных исполняемых файлов. Многие гипертекстовые страницы включают в себя код на языках Java и JavaScript, предназначенный для выполнения в браузере клиента. Более того, часто на сайтах можно встретить файлы, предназначенные к исполнению специальными модулями расширения, встраиваемыми в браузеры[1].

Таким образом, как сервер, так и клиентская программа в нынешних реалиях оказываются скорее платформами для запуска программ, нежели традиционными средствами хранения и визуализации информации.

Часто возникают задачи, связанные с автоматизированным (без непосредственного участия человека) получением информации по протоколу HTTP. К таким задачам относятся, например, зеркалирование сайтов, автоматический поиск информации (data mining), построение баз данных поисковых серверов и т.п. Все эти случаи объединяются возникновением необходимости в клиентской программе, не предназначенной для непосредственной работы с пользователем; в частности, такой программе не нужна визуализирующая подсистема.

Для решения задач такого класса разработана масса программ, каждая из которых имеет весьма узкую область применения. Так, зачастую возможностей флагов командной строки программы *Wget* [8] не хватает для организации пакетной загрузки с сайта именно нужной пользователю информации; приходится либо загружать, наряду с нужными, много ненужных файлов, либо смиряться с необходимостью множества ручных запусков программы.

Это приводит к идее создания универсальной клиентской программы для пакетного взаимодействия по протоколу HTTP, поведение которой задавалось бы программой на встроенном алгоритмически полном языке программирования. Такая программа может быть полезна при решении практически любых задач, связанных с неинтерактивной (пакетной) загрузкой информации из Всемирной паутины.

В некоторых случаях, однако, для решения задачи может попросту не хватить одной только клиентской программы; так происходит, например, при недостатке понимания деталей взаимодействия между сервером и неким исполняемым кодом внутри браузера. Некоторые сайты

[1] Нельзя не отметить, что запуск любого исполняемого кода, полученного по сети, чреват серьёзными проблемами, связанными с безопасностью клиентской машины; практика показывает, что практически любой интерпретатор алгоритмически полного кода, сколь бы тщательно его авторы ни подходили к вопросам защиты, на поверку оказывается уязвим к взлому со стороны исполняемой им программы.

в сети Интернет намеренно организованы таким образом, чтобы затруднить получение информации с них иначе как в интерактивном режиме. Само по себе это вполне нормально, но в некоторых случаях владельцы сайтов вынуждают своих посетителей многократно скачивать одну и ту же информацию, весьма «тяжелую» по объему (чаще всего это видео- и аудиофайлы), порождая, таким образом, бессмысленную нагрузку на каналы передачи данных. Разобраться в происходящем можно, если «встроиться» во взаимодействие между браузером и сервером. Такое «встраивание» предусмотрено протоколом HTTP; соответствующие программы называются прокси-серверами. Прокси-сервер, получив запрос от клиентской программы, передаёт его (уже от своего имени) серверу, а полученный ответ, опять-таки, передаёт клиенту. Прокси-серверы используются во многих ситуациях, в частности, для кеширования, для защиты клиенских программ от нежелательной информации и т.п. Следует отметить, что и мощности конфигурационных файлов современных прокси-серверов часто не хватает для решения возникающих у пользователей специфических задач, и круг таких задач никоим образом не ограничивается анализом взаимодействия клиента и сервера.

Из вышесказанного можно сделать вывод о потенциальной полезности программы, способной играть роль прокси-сервера и управляемой, опять-таки, встроенным интерпретатором алгоритмически полного языка. Поскольку такая программа неизбежно должна уметь играть роль как сервера, так и клиента, возникает логичная идея объединить в одной программе как возможности программируемого клиента, упоминавшегося выше, так и возможности программируемого прокси-сервера. Поскольку с протокольной точки зрения прокси-сервер почти не отличается от обычного HTTP-сервера, в программе имеет смысл предусмотреть возможность использования её и в качестве конечного HTTP-сервера, поскольку это технически достигается сравнительно просто, а по своим возможностям может оказаться полезным (хотя бы для организации управления работой уже запущенной программы через браузер пользователя).

В рассматриваемой реализации роль встроенного интерпретатора играл интерактивный интерпретатор усечённого диалекта языка Лисп, входящий в библиотеку *InteLib* [1]. Это позволило использовать в качестве основного языка реализации язык C++.

2 Структурные элементы веб-агента и их свойства

2.1 Диспетчер взаимодействий

Центральный объект, осуществляющий управление всеми процессами в системе, будем называть диспетчером взаимодействий. Диспетчер взаимодействий осуществляет непосредственное управление клиентскими сессиями и слушающими серверами, отвечает за переключение задач клиентских сессий и за буферизацию данных об общем состоянии системы. Также в функции диспетчера входит приём запросов на вычисление от анализатора пользовательского ввода.

2.2 Клиентские сессии

В системе веб-агента различаются два типа клиентских сессий веб-агента: стандартные сессии и сессии с явным управлением функциями обратного вызова. В первом случае клиент получает задачу на загрузку страницы и выполняет эту задачу асинхронно относительно основного потока вычислений управляющей программы. При этом загруженные данные целиком сохраняются на клиенте вплоть до его удаления. По окончании выполнения задания такой клиент инициирует событие, сигнализирующее о готовности данных. Это событие буферизуется диспетчером до явного запроса события управляющей программой. После этого пользовательская программа может запросить у клиентской сессии загруженное HTTP-сообщение или отдельную его часть. Поскольку стандартные клиентские сессии сохраняют загруженные данные целиком, их не следует использовать для загрузки заведомо больших ресурсов.

Для более гибкого управления предусмотрен еще один тип клиентских сессий — это сессии, управляемые функциями обратного вызова. Такой клиентской сессии, кроме адреса загружаемого ресурса, передается функция для обработки получаемых данных. Эта функция будет вызываться веб-агентом каждый раз при получении клиентом порции данных. Значение, возвращаемое функцией обратного вызова, игнорируется, поэтому смысл самого вызова

заключается только в побочном эффекте. В качестве параметров функции обратного вызова используются объект клиентской сессии и полученная порция данных в виде строки или в виде неструктурированных данных (в последнем случае используется тип `SExpressionRawBuffer` библиотеки *InteLib*). Возможность передачи функции обратного вызова других параметров не предусмотрена, так как все необходимые значения можно поместить в лексический контекст функции, который будет доступен в момент вычисления. Клиентские сессии, управляемые функциями обратного вызова, по умолчанию сохраняют все получаемые данные, поэтому для них также определены функции получения загруженного HTTP-сообщения в структурированном виде. Однако предусмотрена возможность явного сброса буферов клиентской сессии управляющей программой. После вызова этой функции запросы управляющей программой целого HTTP-сообщения будут возвращать ошибку, но доступ к данным заголовка полученного сообщения по-прежнему будет возможен.

Очевидно, что возможности управляемых клиентских сессий включают в себя возможности стандартных сессий: стандартная сессия работает так, как управляемая сессия с «пустой» функцией обратного вызова. Стандартные сессии были введены для оптимизированного выполнения тривиальных задач клиентских сессий и для упрощения работы пользователя с такими задачами.

Ниже приведены простейшие примеры использования стандартных и управляемых клиентских сессий. В обоих случаях результатом программы является загрузка страницы и вывод ответа сервера (вместе с заголовочной частью) на экран.

В случае стандартного клиента управляющая программа получает доступ к данным только по окончании загрузки:

```
(let
    (
         (cl (create-client-by-url "http://someserver/shorttext.txt"))
    )
    (wait-for-event cl) ; блокируется только данный вычислительный поток
    (princ (get-raw-response cl))
    (remove-client cl)
)
```

В случае управляемого клиента можно выводить данные по частям, сразу при получении:

```
(let
    (
        (cl (create-specific-client-by-url
                "http://someserver/longtext.txt"
                #'(lambda (session data) (princ data))
                   ; функция будет вызываться при получении
                   ; каждой порции данных
            )
        )
    )
    (wait-for-event cl) ; блокируется только данный вычислительный поток
    (remove-client cl)
)
```

Для клиентских сессий обоих типов предусмотрена возможность явного задания управляющей программой HTTP-запроса.

2.3 Слушающие сервера

Еще одна часть функциональности веб-агента реализуется слушающими серверами. С каждым слушающим сервером веб-агента связывается функция языка Лисп. Слушающие сервера веб-агента принимают HTTP-запросы от внешних клиентов и запускают вычисление соответствующей Лисп-функции. При этом в случае получения заведомо некорректного запроса или при возникновении ошибки вычислений информирование клиента об ошибке становится ответственностью самого сервера, а не управляющей функции. В этих случаях по умолчанию сервер возвращает следующие коды ошибок:

- `400 - "Bad Request"`, если формат запроса не соответствует протоколу HTTP;
- `415 - "Unsupported Media Type"`, если содержимое запроса типа `POST` представляет собой бинарные данные;
- `505 - "HTTP Version Not Supported"`, если запрос соответствует версии протокола, отличной от HTTP/1.1;
- `500 - "Internal Server Error"`, если при вычислении функции произошла ошибка: исключительная ситуация библиотеки *InteLib* или веб-агента.

В системе предусмотрена возможность изменения логики действий слушающего сервера в случае возникновения таких ошибок. Для этого управляющая программа может определить для слушающего сервера функции обработки ошибок.

Функция, связываемая с сервером при его создании, должна принимать два аргумента. Это требование обосновано соглашением о передаче параметров: функции, вычисляемые слушающими серверами, в качестве параметров получают объект соответствующей сессии и HTTP-запрос. Даже если связываемая с сервером функция не использует эти значения, она должна принимать на вход соответствующие параметры.

2.4 События и ошибки в системе веб-агента

Помимо задачи перенаправления вызовов клиентам и серверам, в функции диспетчера также входит управление очередями событий веб-агента и передачей управляющей программе информации о возникающих ошибках. События и ошибки веб-клиента содержательно бывают очень похожи, но они имеют разное происхождение, на котором следует остановиться подробнее. События веб-агента — это объекты, сохраняющие информацию об изменениях в системе, которые могут быть интересны управляющей программе и которые произошли *асинхронно* относительно обращений пользовательской программы к веб-агенту. Событиями в системе могут быть, например, окончание загрузки требуемой страницы клиентом, или возникновение некорректных ситуаций при работе клиента или сервера. Ошибки же, напротив, всегда появляются *синхронно* с обращениями управляющей программы к веб-агенту и возвращаются в ответ на эти обращения. Ошибка может возникнуть, например, при попытке прикладной программы обратиться к клиенту, который был удален. Для событий веб-агента реализован механизм буферизации, для ошибок необходимости в буферизации нет.

2.5 HTTP-сообщения

В системе веб-агента предусмотрены средства для структурированного представления сообщений протокола HTTP. Как на клиентской, так и на серверной стороне производится проверка корректности формата получаемых сообщений и разбор сообщений на структурные составляющие. В языке управления веб-агентом предусмотрена возможность получения от клиентской сессии отдельных структурных составляющих сообщения — статуса, заголовочной части или отдельного поля заголовка, тела сообщения. При этом получение данных из заголовка возможно до окончания загрузки тела сообщения.

2.6 Модуль вычислений

Важнейшей частью веб-агента является модуль вычислений. С точки зрения диспетчера этот модуль является абстрактным вычислителем, и набор его функций не зависит от природы проводимых вычислений. Конкретная реализация этого модуля как Лисп-вычислителя использует механизмы, предоставляемые библиотекой *InteLib*.

Один и тот же модуль вычислений используется как для основного потока вычислений, так и для вычислений, связанных с запросами к слушающим серверам. При этом все вычисления ведутся в едином глобальном контексте.

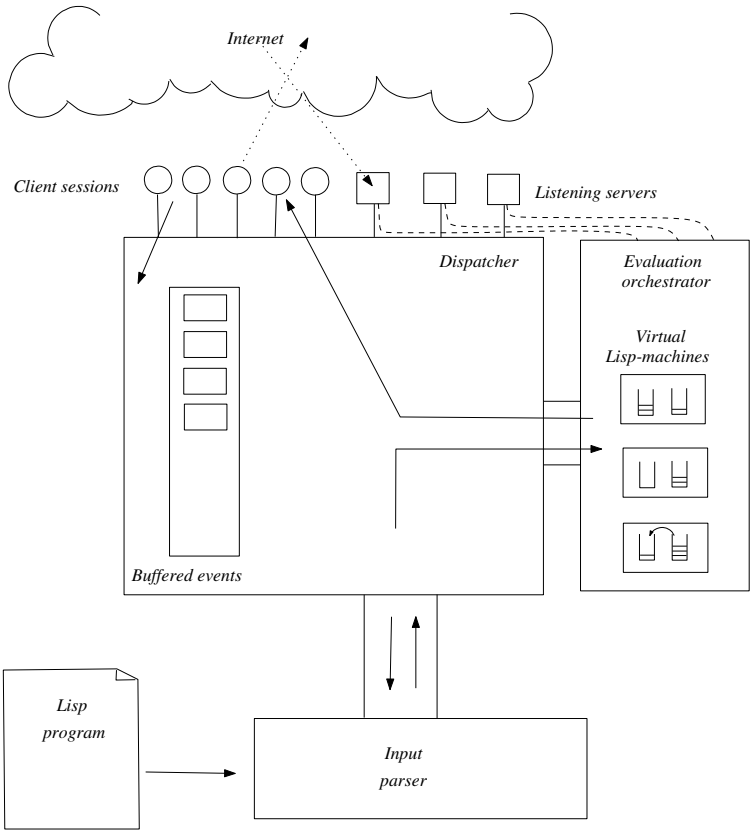


Рис. 1: Структурное представление веб-агента

2.7 Анализатор пользовательского ввода

Идея, на которой основан принцип функционирования веб-агента, подразумевает, что задержка поступления данных от анализатора пользовательского ввода не должна блокировать выполнение текущих задач веб-агента. Это условие достигается путем выделения анализатора ввода интерактивного интерпретатора в отдельный процесс. В таком случае в ответственность анализатора пользовательского ввода входит только поддержка возможностей редактирования вводимой строки, выделение в потоке ввода форм языка Лисп и передача полученных данных процессу, в котором работает диспетчер. Перевод же данных в структуры, передаваемые Лисп-машине, происходит уже в процессе диспетчера. При реализации этой возможности использовались средства библиотеки *InteLib* для интерактивного ввода программы на Лиспе, которые в свою очередь используют API библиотеки *GNU Realine*.

3 Логическая структура системы

Рассмотрим структурные связи между компонентами веб-агента. Центральным элементом системы веб-агента является диспетчер взаимодействий; именно он управляет информационными потоками между пользовательской программой и клиентскими сессиями и серверами, а также хранит информацию о происходящих событиях. У всех слушающих серверов есть прямые ссылки на модуль вычислений, поскольку процедура обращения сервера к вычислителю проста, и перегружать диспетчер этой задачей необходимости нет. Аналогичные ссылки могут быть и у клиентских сессий, если для них явно заданы функции реакции на приход данных. Схема логической структуры веб-агента показана на рисунке 1. В целом такая схема зависимостей соответствует общепринятым канонам объектно-ориентированного проектирования [4].

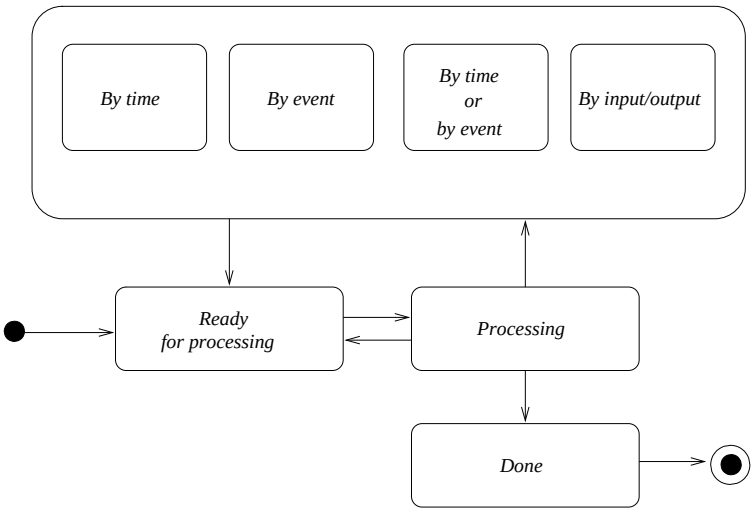


Рис. 2: Диаграмма состояний вычислительных потоков веб-агента

4 Принципы функционирования веб-агента

4.1 Потоки вычислений в системе веб-агента

Функционирование клиентских сессий и слушающих серверов происходит асинхронно относительно друг друга и относительно работы анализатора пользовательского ввода. При этом в системе одновременно сосуществуют несколько потоков Лисп-вычислений, выполняющихся псевдопараллельно. Можно выделить четыре типа вычислительных потоков:

- основной поток управляющей программы;
- потоки вычислений слушающих серверов;
- потоки вычислений функций реакции на приход данных клиентских сессий;
- потоки обработки ошибок слушающих серверов.

Для каждого из этих потоков определен набор состояний, в которых может находиться поток. Диаграмма состояний вычислительных потоков приведена на рисунке 2. Стрелками на диаграмме обозначены возможные переходы между состояниями. Отметим, что в состоянии вычисления в каждый момент времени может находиться не более чем один вычислительный поток.

4.2 Переключение вычислительных потоков

Переключение вычислительных потоков осуществляется в трех случаях:

- По окончании вычисления потока. Сразу по окончании вычисления поток удаляется.
- При вызове функций, переводящих поток в состояние блокировки. Это могут быть функции ожидания события, функции блокировки на заданный интервал времени или функции ввода-вывода.
- При превышении потоком количества шагов виртуальной машины, разрешенного для непрерывного вычисления. Это сделано во избежание дискриминации вычислительных потоков.

Дескрипторы сокетов и потоков ввода/вывода контролируются системным вызовом `select` в цикле работы диспетчера взаимодействий. На каждой итерации цикла диспетчер обрабатывает активизацию дескрипторов и передает модулю вычислений данные об изменении состояния системы. Это могут быть события, сигнализирующие об окончании загрузки данных клиентскими сессиями или о произошедших ошибках, а также активизация дескрипторов ожидающих потоков ввода/вывода. На основании этих данных модуль вычислений разблокирует соответствующие вычислительные потоки.

События на дескрипторах, контролируемых вызовом `select`, могут также вызывать создание новых вычислительных потоков. Это происходит, например, в случае окончания получения запроса слушающим сервером, когда запускается вычисление соответствующей серверу функции.

После обработки активизации дескрипторов, диспетчер вызывает метод `PerformStep` объекта модуля вычислений, запуская тем самым очередной шаг вычислений для готовых вычислительных потоков. Логика выполнения шага вычислений для каждого отдельного вычислительного потока зависит от конкретной реализации вычислительных потоков. В системе веб-агента, управляемого на языке Лисп, выполнение вычислительных потоков осуществляется виртуальными Лисп-машинами (более подробно см. в подразделе 4.3).

После выполнения вычислений диспетчер запрашивает у модуля вычислений информацию, необходимую для определения максимального времени ожидания вызова `select`. В случае если существуют вычислительные потоки, готовые к выполнению, это время устанавливается в ноль. Если готовых вычислительных потоков нет, но есть потоки, которые могут быть разблокированы по времени, передается минимальное время ожидания для таких потоков. Если вычисления по всем потокам завершены или заблокированы по событию, время ожидания вызова `select` не ограничивается.

4.3 Многопоточный Лисп-вычислитель

В реализации абстрактного модуля вычислений как многопоточного Лисп-вычислителя отдельным потокам соответствуют разные виртуальные Лисп-машины, работающие в едином глобальном вычислительном контексте. Эти Лисп-машины представляют собой объекты класса `WeblispContinuation`, наследуемого, с одной стороны, от класса `LispContinuation` библиотеки *InteLib*, а с другой — от абстрактного класса `VirtualMachine`, представляющего вычислительный поток. Таким образом, объекты этого класса способны производить вычисления в соответствии с моделью языка Лисп, и блокировать вычисления в соответствии с логикой модуля вычислений веб-агента.

Лисп-вычислитель библиотеки *InteLib* представляет собой виртуальную вычислительную машину, в которой вычисление формы языка Лисп разделяется на отдельные атомарные шаги. При этом вычисление может быть прервано после любого из таких атомарных шагов и в дальнейшем возобновлено без потери уже вычисленных данных. Это свойство позволяет программе переходить в состояние отложенных вычислений (так называемые *continuations* [5]). Возможность прерывания Лисп-вычислений используется и виртуальными Лисп-машинами веб-агента для ограничения количества шагов, которые может выполнить в непрерывном режиме один вычислительный поток.

Идея многопоточного Лисп-вычислителя привносит в реализацию дополнительные сложности, связанные с возможными взаимдействиями вычислительных потоков. В частности, в случае если несколько вычислительных потоков используют и модифицируют значение некоторой глобальной переменной, может возникнуть необходимость в атомарности таких операций чтения/записи. Для корректной обработки такого рода ситуаций необходимо предусмотреть механизм, позволяющий пользовательской программе устанавливать запрет переключения вычислительных потоков в некоторой секции. В нынешней реализации эта задача пока не решена.

5 Поддержка протокола HTTP

В рамках поддержки веб-агентом протокола HTTP версии 1.1 (стандарт RFC 2616 — [6]) при работе с HTTP-сообщениями поддерживаются следующие возможности:

- разделение сообщения на начальную строку, заголовок и содержимое путем выделения парных символов CR и LF[2] (согласно формату сообщений HTTP/1.1: см. [6], п. 4.1);
- обаботка ведущих и конечных пробелов в значениях заголовков, обработка переносов строк отдельного поля заголовка согласно [6], п. 4.2);
- определение длины сообщения по значению поля `Content-Length` в заголовке или на основе кодировки `"Transfer-Encoding: chunked"` ([6], п. 3.6.1); если эти поля не заданы, предполагается разрыв соединения сервером по окончании передачи сообщения;
- требование обязательного наличия в заголовке поля `Host` во всех запросах к слушающим серверам (согласно [6], п. 14.23).

Веб-агент позволяет пользовательской программе как явно задавать запросы, посылаемые клиентскими сессиями внешним серверам, так и использовать запросы стандартного вида. В заголовке запроса стандартного вида задаются поля `Host`, `Content-Length` и `User-Agent` (последний — со значением `Weblisp/0.1`).

Ответы слушающих серверов на запросы внешних клиентов имеют в заголовке следующие поля: `Content-Type`, `Content-Length`, `Date`, `Last-Modified` (с датой последнего обращения к Лисп-вычислителю), `Connection` (со значением `close`, так как сервер разрывает связь после ответа).

6 Задача перехвата видео- и аудиопотоков

Задача оптимизации Интернет-трафика при повторном обращении к ресурсам решается всеми современными браузерами путем кэширования веб-страниц, но гибкая настройка такого кэширования для конкретных задач на уровне браузера невозможна. Например, если существует потребность многократного просмотра некоторого видеофайла, получаемого из сети, то естественно сохранить этот файл на прокси-сервере вместе с HTTP-заголовком и при повторном обращении не запрашивать файл из сети, а доставать его из репозитория.

Такая задача естественным образом решается с помощью веб-агента. В качестве демонстрационного примера была написана программа на языке Лисп для управления работой прокси-сервера, основной задачей которого является организация репозитория видеофайлов, полученных с сайта *www.youtube.com*. В этой программе прокси-сервер анализирует запрос клиента и выполняет следующие действия:

- Если браузер осуществляет запрос по адресу *http://__repository/*, то в браузер передается список видеофайлов, сохраненных в репозитории.
- Если целью запроса не является видеофайл с сайта *www.youtube.com*, прокси-сервер просто передаёт запрос клиента серверу и возвращает клиенту получаемый ответ.
- Если целью запроса является видеофайл с сайта *www.youtube.com*, и данный файл есть в репозитории, клиенту возвращается HTTP-заголовок и содержимое видеофайла, найденное в репозитории.
- Если целью запроса является видеофайл с сайта *www.youtube.com*, и данного файла нет в репозитории, прокси-сервер передает запрос клиента на сервер, а получаемый ответ передает клиенту и сохраняет в репозитории.

[2] US-ASCII CR, carriage return (13) и US-ASCII LF, linefeed (10)

7 Перспективы дальнейшей работы

В качестве ближайших перспектив описываемой разработки можно выделить оптимизацию существующего аппарата управления потоками Лисп-вычислений, а также разработку минимальных средств взаимодействия для вычислительных потоков. Также для корректной работы веб-агента на сложных сценариях с совместным использованием файлов и перекрестными блокировками вычислительных потоков потребуется разработка механизма разрешения тупиков.

Для упрощения программирования веб-агента имеет смысл снабдить его набором базовых функций работы с документами в форматах HTML/XML, в частности, функциями выбора ссылок и представления XML-деревьев в виде S-выражений. В нынешней реализации всё это остается задачей пользовательской программы.

Список литературы

[1] И.Г. Головин, А.В. Столяров. Объектно-ориентированный подход к мультипарадигмальному программированию // Вестник Московского Университета, сер. 15 вычисл. матем. и киберн. 2002. N 1. С. 46–50.

[2] Хювёнен Э., Сеппянен Й. Мир Лиспа. В 2-х т. Т. 1: Введение в язык Лисп и функциональное программирование. М.: Мир, 1990. 458 с.

[3] W. Richard Stevens. UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI, Prentice Hall, 1998. 1240 p.

[4] Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, второе изд. М.: Бином, СПб.: Невский диалект, 1998. 560 с.

[5] John C. Reynolds. The discoveries of continuations // Lisp and Symbolic Computation, 1993. N 6(3-4). P. 233–248.

[6] Hypertext Transfer Protocol – HTTP/1.1. Official Internet Protocol Standards: Standards Track: RFC 2616.

[7] The InteLib home page [HTML] (`http://www.intelib.org/`).

[8] GNU Wget Official Site [HTML] (`http://www.gnu.org/software/wget/`).

УДК 551.501 + 551.46

АЛГОРИТМЫ ИНТЕРПОЛЯЦИИ И ЭКСТРАПОЛЯЦИИ ОПЕРАТИВНЫХ ГЕОФИЗИЧЕСКИХ ДАННЫХ НАБЛЮДЕНИЙ

© 2009 г. Н. Б. Захарова, С. А. Лебедев

lebedev@wdcb.ru

Кафедра вычислительных технологий и моделирования; Геофизический центр РАН

Введение

В последние годы непрерывно возрастает интерес к задачам ассимиляции (усвоения) и обработки данных наблюдений. Эти задачи фактически представляют собой обратные задачи [1] (или задачи управления) для сложных математических моделей, описываемых нелинейными уравнениями в частных производных. А процессы ассимиляции данных наблюдений являются процедурами замыкания данных задач. Наибольшие приложения они получили в метеорологии и океанологии, где данные измерений основных физических полей атмосферы и океана ассимилируются в математических моделях с целью получения начальных условий (или других параметров модели) или с целью более адекватного воспроизведения реальных физических процессов или прогноза состояния моделируемой среды.

Среди методов вычислительной математики, применяемых для решения задач усвоения данных наблюдений, значительную роль играют методы их предварительной обработки, анализа и алгоритмы интерполяции и экстраполяции, которые используются для реализации одного из основных этапов решения задач ассимиляции данных. В связи с этим, разработка таких алгоритмов и соответствующего программного обеспечения, основанных на современных подходах и учитывающих последние достижения в этом направлении, является актуальной проблемой.

В настоящей работе описаны оперативные данные измерений характеристик вод Мирового океана, представлено описание нескольких методов интерполяции данных наблюдений и продемонстрированы результаты некоторых численных экспериментов, проведенных с помощью Комплекса программ, разработанным авторами для использования в задачах усвоения оперативных геофизических данных наблюдений.

1 Оперативные данные о состоянии Мирового океана

Основными параметрами, характеризующими состояние Мирового океана, являются: температура, соленость, скорость и направление течений, уровень морской воды, содержание кислорода, содержание нитратов, содержание фосфатов и т.д. [2, 3]. До середины прошлого столетия эти параметры измерялись традиционными контактными методами измерений. С развитием приборостроения появились дистанционные методы зондирования поверхности океана как с борта самолета, так и искусственных спутников Земли (ИСЗ). Таким образом, данные о состоянии океана условно можно подразделить на два типа: контактные и дистанционные, оперативность поступления которых в несколько раз превышает поступление данных, измеряемых контактными методами.

Современные методам дистанционного зондирования Земли из космоса подразделяют на три типа [4]: пассивные, полуактивные и активные. Пассивные методы основаны на регистрации теплового излучения (ИК и СВЧ), видимого излучения и естественного гамма-излучения с подстилающей поверхности. Полуактивные методы основаны на облучении естественными и искусственными источниками электромагнитного излучения в широком спектральном диапазоне и в анализе спектрального состава принятого сигнала поверхности акватории. При использовании активных методов исследуемая водная поверхность облучается источниками излучения заданного спектрального состава с регистрацией или отраженного излучения, или флуоресценции, или комбинационного рассеяния.

Температура поверхности океана является одним из первых океанографических параметров, которые начали измерять с борта ИСЗ. Она рассчитывается по данным о радиояркостной температуре, измеряемой ИК и СВЧ-радиометрами (включая и радиометры сканирующие вдоль подспутникового следа), и по данным сканеров видимого диапазона, которые имеют дополнительный канал в ИК-диапазоне. Обработка данных радиометром и сканеров проводится в несколько этапов. Самым существенным из них является привязка к географической сетке. Эта процедура достаточно трудоемка, поэтому для решения задачи ассимиляции данных необходимо использовать спутниковую информацию, подвергнутую первичной обработке.

Высота морской поверхности океана рассчитается по данным спутниковой альтиметрии в точке вдоль подспутникового следа. Сама высота морской поверхности не может использоваться в задаче ассимиляции данных. Для этого необходимо рассчитать динамическую топографию, которая определяется как отклонение высоты морской поверхности от поверхности геоида (эквипотенциальной поверхности).

В настоящее время проводятся отдельные эксперименты по разработке алгоритмов и приборов расчета солености морской воды на поверхности по данным дистанционного зондирования.

Точность расчета этих параметров представлена в Таблице 1.

Датчик			**Точность**
Тип	**Название**		
Температура поверхности океана			
ИК-радиометр	Advanced Very High Resolution Radiometers	AVHRR	0.3–0.5 ° C
Сканер видимого диапазона	Moderate-resolution Imaging Spectroradiometer	MODIS	0.3 ° C
ИК-радиометр	Along-Track Scanning Radiometer	ATSR	0.3 ° C
СВЧ-радиометр	Special Sensor Microwave Imager	SSMI	0.7 ° C
Высота морской поверхности			
Альтиметр	Radio Altimeter	Poseidon - 2	4.2 см

Таблица 1: Точности измерения основных океанографических параметров с борта ИСЗ.

Профилирующие буи ARGO представляют собой смешенный тип измерения основных океанографических параметров. Профили температуры и солености (или электропроводность) измеряются контактными методами, но сами данные передаются с помощью ИСЗ. Таким образом они дают уникальную оперативную информацию о состоянии океана.

Спущенный на воду буй дрейфует на поверхности в течение некоторого промежутка времени, достаточного для передачи данных на проходящие спутники (обычно 6-12 часов). После опускания на заданный горизонт буй дрейфует в течение 10 дней, после чего поднимается на поверхность с постоянной скоростью 1 м/с, проводя измерения давления, температуры и электропроводности, на основании которой в дальнейшем рассчитывается соленость морской воды (для горизонта 2000 м этот процесс обычно занимает около 6 часов), и с поверхности передаёт информацию на ИСЗ. Цикл повторяется до тех пор, пока не истощатся батареи или буй не будет выловлен рыбаками. Поймать буй специально довольно сложно — требуется сочетание многих условий: спутниковый телефон, пеленгационный контур и хорошая погода.

Целью проекта ARGO создание и поддержание глобальной сети из 3000 буев, что должно соответствовать более 100000 измерений различными зондами. Задачами проекта являются: размещение в Мировом океане заданного количества буев; создание национальных центров ARGO; обеспечение свободного доступа к данным; передача в Глобальную сеть данных в течение 24 часов (требуемых на первичный контроль качества) с момента поступления.

О востребованности информации с профилирующих буев ARGO говорит тот факт, что число корабельных измерений различными зондами сокращается по сравнению с измерениями данного типа буев.

Подготовка данных о профилях температуры и солёности с буёв ARGO для применения их в задаче ассимиляции данных сама по себе является достаточно сложной задачей. Она реа-

лизуется в два этапа. На первом этапе проводится интерполяция данных на расчетные уровни модели, на втором этапе – интерполяция на расчетную сетку модели по "горизонтальным переменным". В значительной степени целью настоящей работы было рассмотреть ряд алгоритмов решения этих двух этапов и реализации их в соответствующем комплексе программ.

2 Алгоритмы интерполяции и экстраполяции

В данном разделе представлено несколько методов интерполяции (как одномерной, так и двумерной) данных наблюдений.

2.1 Линейная интерполяция

Линейная интерполяция — интерполяция алгебраическим двучленом $P(z) = az + b$ функции одной переменной $f(z)$, заданной в двух точках z_0 и z_1 отрезка $[a, b]$ [5]. Геометрически это означает замену графика функции f прямой, проходящей через точки $(z_0, f(z_0))$ и $(z_1, f(z_1))$. Уравнение такой прямой имеет вид:

$$\frac{P(z) - f(z_0)}{f(z_1) - f(z_0)} = \frac{z - z_0}{z_1 - z_0},$$

отсюда для $z \in [z_0 - z_1]$

$$f(z) \approx P(z) = f(z_0) + \frac{f(z_1) - f(z_0)}{z_1 - z_0}(z - z_0)$$

Это и есть формула линейной интерполяции, при этом

$$f(z) = P(z) + R(z),$$

где $R(z)$ — погрешность формулы:

$$R(z) = \frac{f"(\psi)}{2}(z - z_0)(z - z_1), \quad \psi \in [z_0, z_1].$$

Алгоритм линейной интерполяции является одним из самых простых методов интерполяции данных. Этот алгоритм может быть использован для интерполяции данных наблюдений, полученных с буёв ARGO по вертикальной переменной (“по глубине”).

2.2 Оптимальная линейная экстраполяция (интерполяция) данных

Метод оптимальной линейной экстраполяции [6]-[8] является математической основой статистического прогнозирования, суть которого состоит в прогнозе будущего значения параметра состояния системы на основе его значений в прошлом и известных статистических связей между этими значениями.

Пусть на некотором промежутке $[a, t]$ изменения аргумента, предшествующем моменту времени t , имеется реализация $f(t)$ случайного процесса $F(t)$, статистические характеристики которого известны. Требуется дать прогноз значения этой реализации $f(t + T)$ в некоторый последующий момент $t + T, T > 0$. Величина T называется упреждением.

Поскольку мы имеем дело со случайными процессами, то нас интересует нахождение такого способа решения задачи, который давал бы наилучший в некотором смысле результат по всему множеству реализаций, т.е. нахождение такого оператора, который в применении к множеству реализаций $f(t)$ давал бы наилучшее в некотором смысле значение $f(t + T)$.

Обозначив искомый оператор через L , можем записать

$$F(t + T) = L[F(t)]. \tag{1}$$

Обозначим через δ разность между истинным значением F(t+T) и значением, полученным по формуле (1). Наиболее удобным с математической точки зрения критерием качества экстраполяции является обращение в минимум математического ожидания квадрата разности

$$M[\delta^2] = M\{[F(t+T) - L[F(t)]]^2\}. \quad (2)$$

Оператор L, при котором выражение (2) обращается в минимум, называют оптимальным оператором, а формулу (1) – формулой оптимальной экстраполяции.

В последние годы метод оптимальной линейной интерполяции применялся для прогноза различных гидрометеорологических параметров. При этом, по-видимому, не следует противопоставлять его другим методам прогнозирования, в частности динамическим методам, основанным на использовании уравнений гидродинамики, а стараться сочетать различные методы прогноза для получения более надёжных результатов. Одним из путей такого сочетания является получение так называемого "взвешенного" прогноза. Допустим, что получены прогнозы одного и того же элемента двумя различными методами. Первый метод даёт прогнозируемое значение z_1, второй - z_2. Тогда в качестве прогнозируемого значения можно использовать величину $z = pz_1 + qz_2$, где веса p и $q = (1-p)$ выбираются из тех или иных разумных соображений. В частности, если известны дисперсии ошибок каждого метода прогноза, то веса можно выбирать как величины, обратно пропорциональные этим дисперсиям.

2.3 Метод простейшей интерполяции данных измерений

Следующий алгоритм реализует простейшую интерполяцию, переводит данные измерений из хаотичного многообразия точек, в которых они известны, на регулярную сетку путём усреднения данных в узлах, используя все точки, к которым данный узел является ближайшим [9].

В данном алгоритме для каждого (i,j)-го узла регулярной сетки рассматривается дополнительная подобласть Ω_{ij}.

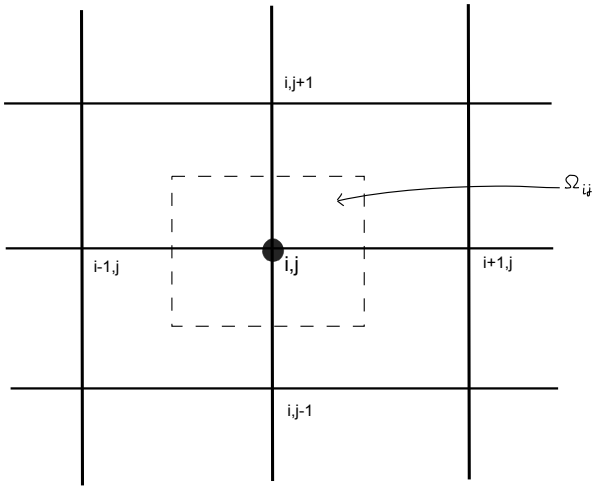


Рис. 1: Область влияния узла (i,j) расчетной сетки

Интерполяционное значение в этом узле считается с помощью данных измерений всех точек, входящих в данную подобласть по следующей формуле:

$$Z_{ij} = \frac{1}{m}\sum_{k=1}^{m} \hat{Z}_k,$$

где $\hat{Z}_k$ - значение данных измерений в точке $(\hat{x}_k, \hat{y}_k) \in \Omega_{ij}$, $m \equiv m(i,j)$ - число точек, попавших в подобласть Ω_{ij}, в которых имеются данные наблюдений, Z_{ij} - значение температуры в узле регулярной сетки с координатами (x_i, y_j).

Условие попадания точки с координатами $(\hat{x}_k, \hat{y}_k)$ в подобласть Ω_{ij}:

$$|x_i - \hat{x}_k| < \frac{1}{2}h_x, \quad |y_j - \hat{y}_k| < \frac{h_y}{2},$$

где h_x, h_y - шаги сетки по x и по y соответственно.

Описанный метод также относится к классу простейших алгоритмов интерполяции данных наблюдений. В отличие от него, метод обратных взвешенных расстояний относится к группе алгоритмов, передающих значения экспериментальных величин более точно.

2.4 Метод обратных взвешенных расстояний

Метод обратных взвешенных расстояний [9] основан на вычислении весовых коэффициентов, с помощью которых взвешиваются значения данных измерений $(\hat{Z}_k)$ в точках наблюдений при построении интерполяционной функции. Вес, присвоенный отдельной точке данных при вычислении значения в узле сетки, пропорционален заданной степени обратного расстояния от точки наблюдения до узла сетки. При вычислении интерполяционной функции в какомто узле сетки сумма всех назначенных весов равна единице, а весовой коэффициент каждой точки, где проводились измерения, является долей этого общего единичного веса. Если точка наблюдения совпадает с узлом сети, то весовой коэффициент этой точки полагается равным единице, а всем другим наблюденным точкам присваиваются нулевые веса. Другими словами, в этом случае узлу сетки присваивается значение соответствующего наблюдения, и, следовательно, данный метод работает как точный интерполятор.

Формула интерполяции, используемая в данном алгоритме, имеет вид:

$$Z_{ij} = \frac{\sum\limits_{k=1}^{m} \frac{\hat{Z}_k}{r_{ijk}^{\beta}}}{\sum\limits_{k=1}^{m} \frac{1}{r_{ijk}^{\beta}}},$$

где Z_{ij} – интерполяционное значение для (i,j)-го узла сетки; $\hat{Z}_k$ – значение данных измерений в k-ой точке наблюдений; r_{ijk} – расстояние между (i,j)-ым узлом и k-ой точкой наблюдений; $m \equiv m(i,j)$ - число точек, учитывающихся при вычислении значения в (i,j)-ом узле , β – степень обратного расстояния.

Представленный в этом разделе метод является одним из составляющих Комплекса программ, разработанного для решения задачи обработки и усвоения спутниковых данных и данных, поступающих с буёв ARGO.

3 Обработка данных, поступающих с буёв ARGO

На основе описанных выше алгоритмов, формулируются методы и процедуры обработки и интерполяции данных наблюдений, поступающих с буёв ARGO, которые составляют теоретическую основу разработанного авторами программного обеспечения. Задача осуществления процесса обработки и интерполяции данных делится на две большие подзадачи. Первая - интерполяция данных наблюдений по вертикальной переменной z (или σ-системы координат), для построения вертикальных профилей температуры и солёности. Сигма координатами $\sigma = \frac{z}{H}$ называется система координат, в которой поверхность моря принята как 0-вая глубина, а дно H, т.е. максимальная глубина (в каждой точке) – равна 1. Вторая - интерполяция по "горизонтальной плоскости по широте и долготе, - для построения полей температуры и солёности на заданных уровнях (глубинах).

3.1 Интерполяции данных наблюдений с целью построения вертикальных профилей температуры в океане

Данные каждой буйковой станции ARGO представляют собой набор данных по вертикальной переменной. Задача интерполяции данных наблюдений с буёв ARGO заключается

в интерполяции данных, разбросанных хаотически по всей глубине измерений, на заданные уровни, что делает эти задачи достаточно сложными для решения.

Рассмотрим два возможных варианта интерполяции данных наблюдений, получаемых с буёв ARGO в зависимости от модели термогидродинамики: интерполяция данных на стандартные горизонты или интерполяция по уровням в σ-системе координат. В связи с этим, существуют два подхода к задаче обработки и интерполяции этих данных наблюдений для построения вертикальных профилей:

(А) Полученные данные наблюдений с буёв ARGO интерполировать на 33 стандартных горизонта [10], а после, при передаче в “прямую модель”, интерполировать на уровни в σ-системе координат.

(Б) Полученные с буёв ARGO данные наблюдений сразу интерполировать на уровни в σ-системе координат.

Проинтерполированные на стандартные уровни, данные наблюдений передаются в модель для дальнейшей обработки, где пересчитываются на σ-уровни, поэтому интерполяция данных наблюдений сразу на уровни в σ-координатах уместна и удобна.

Описанная выше процедура интерполяции данных по вертикальной компоненте реализуется в разработанном Комплексе программ.

3.2 Интерполяция данных наблюдений, поступающих с буёв ARGO, по "горизонтальной плоскости"(по широте и долготе)

Данные наблюдений, поставляемые с буёв ARGO, представляют собой полезную информацию в плане получения оперативных сведений о состоянии вод Мирового океана. Сравнительно малое количество данных с буйковых станций и их хаотическое распределение по пространству недостаточно, чтобы смоделировать полную и точную картину поведения вод в океане. Тем не менее, хочется иметь представление об измеряемых величинах не только в точках измерений буями, но и по всей акватории океана. В связи с этим, возникает задача об интерполяции данных наблюдений по "горизонтальной плоскости"(по широте и долготе).

Можно предложить следующую реализацию алгоритма интерполяции данных, состоящую из двух этапов.

На первом этапе производится интерполяция данных наблюдений с буёв ARGO в узлы сетки методом обратных взвешенных расстояний, описанном в разделе 2 данной работы.

Второй этап - экстраполяция данных по всей области океана,- заключается в осуществляемом в несколько итераций путём пересчёта данных с узлов сетки, где есть значения, на узлы этой же сетки, где значения не известны, с радиусом влияния, охватывающим все ближайшие к данному узлу узлы сетки. После первого этапа интерполяции получаем сетку, в некоторых узлах которой есть значения, и дальше работаем с ней. Используя эту сетку, проверяем все узлы, и если значение в узле неизвестно, то вычисляем его по значениям в узлах, входящих в окружность с радиусом $r_n = n\sqrt{h_x^2 + h_y^2}$, где h_x и h_y – шаги сетки по широте и долготе соответственно, если значения как минимум в двух из них нам известны, n – число итераций. После этого работаем уже с вновь полученной сеткой таким же образом. Процесс длится столько итераций, сколько нужно для заполнения всей сеточной области, соответствующей акватории Мирового океана.

4 Результаты численных экспериментов

В Институте вычислительной математики (ИВМ) РАН разработана первая версия “Информационно-вычислительной системы вариационной ассимиляции для анализа сложных моделей” (ИВС-1) [11]-[14]. Разработанная Информационно-вычислительная система состоит из нескольких подсистем, каждая из которых может рассматриваться как самостоятельная информационно-вычислительная система (для решения “прямой модели”, для инициализации гидрофизических полей в океане, для вариационной ассимиляции поверхностной температуры, для расчета распространения загрязнений в окружающей среде и др.).

Как одна из таких подсистем ИВС-1, в ИВМ РАН совместно с кафедрой Вычислительных технологий и моделирования факультета ВМК МГУ им. М.В.Ломоносова разработана первая

версия Комплекса программ пересчёта и интерполяции данных наблюдений. Он представляет собой основанный на описанных выше алгоритмах комплекс программ для обработки данных, получаемых из базы данных "Мировой океан - ИВМ РАН" [2]-[3], их интерполяции и записи в файлы в заданном формате для дальнейшего усвоения "прямой моделью" в целях прогнозирования.

Способ обработки и пересчёта определяется в зависимости от вида получаемых данных. Так, например, в Комплексе программ реализованы разные методы обработки и интерполяции данных, получаемых со спутников и буйковых станций. Обработанные и проинтерполированные, данные передаются для усвоения в "прямую модель".

Программно Комплекс реализован на языке Fortran [15] и может работать в операционных системах Linux, Unix или Windows. О форматах входной и выходной информации можно посмотреть в работе [16].

Приведём некоторые результаты нескольких численных экспериментов, осуществленных с помощью упомянутого выше Комплекса программ.

Пример 1. Интерполяция данных наблюдений с буёв ARGO по вертикали.

В качестве примера получаемых профилей температуры, приведём несколько изображений выходных данных программы для интерполяции данных наблюдений по вертикали, и покажем несколько результатов численных экспериментов.

Ниже на рис.2 приведено изображение данных температуры, пересчитанных на стандартные уровни по всей измеряемой глубине за 4 января 2004 года в точке с координатами 66.6° в.д. и 9.4° с.ш.

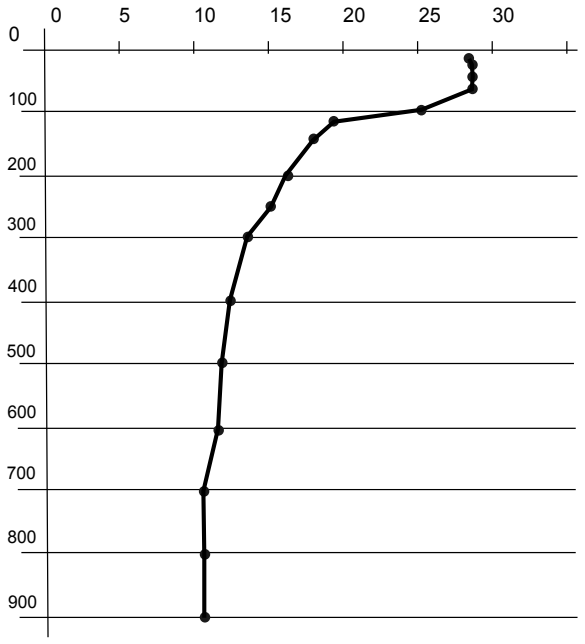


Рис. 2: Профиль температуры за 4 января 2004 года в точке с координатами 66.6° в.д. и 9.4° с.ш.

При интерполяции данных наблюдений сначала на стандартные горизонты, а после на σ-уровни, возникает некоторая ошибка по сравнению с интерполяцией данных наблюдений сразу на σ-уровни. Для сравнения работы этих двух программ по интерполяции данных наблюдений, приведём пример ещё одного численного эксперимента.

На рис. 3 приведено изображение данных наблюдений температуры, измеренных одним из буёв ARGO 14 января 2004 года в Индийском океане в точке с координатами 68.6° в.д.

и 9.0° ю.ш. На графике также отмечены эти данные наблюдений, проинтерполированные на стандартные горизонты.

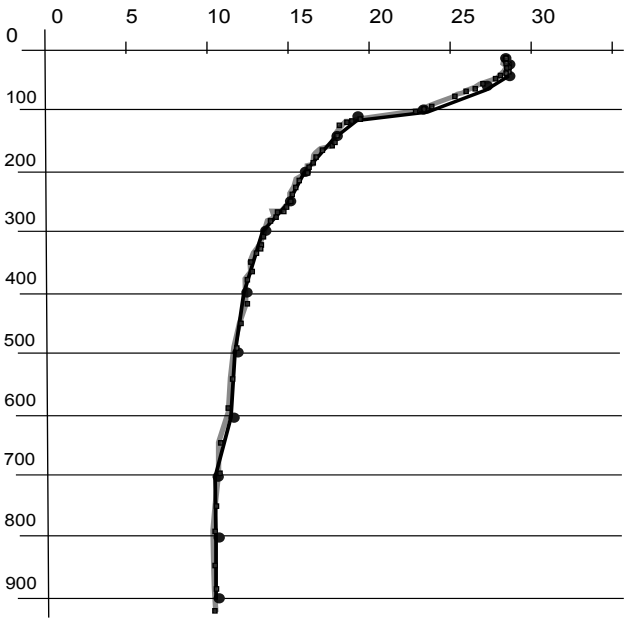


Рис. 3: Данные наблюдений температуры и их интерполяция на стандартные горизонты

Как видно из рис. 3, возникает некоторая погрешность при интерполяции данных наблюдений на стандартные горизонты. В связи с этим возникает разница между данными наблюдений, проинтерполированными на уровни в σ-координатах и данными, проинтерполированными сначала на стандартные горизонты, а после на уровни в системе σ-координат.

На рис. 4 изображены выходные данные работы программы для интерполяции данных на уровни в σ-координатах.

Разными маркерами на рис. 4 обозначены данные наблюдений, проинтерполированные на уровни в σ-координатах, и данные наблюдений, проинтерполированные сначала на стандартные горизонты, а после на уровни в σ-координатах. Видно, что кривые практически совпадают. Однако отличия есть, и для наглядности приведём разницу между двумя графиками выходных данных, изображенными выше (рис. 5).

Как видно из рис. 5, ошибка может составлять на некоторых уровнях до 0.25°C, что соизмеримо с точностью измерений.

Пример2. Интерполяция данных наблюдений с буёв ARGO по горизонтали.

Для иллюстрации работы метода, применяющегося для обработки и интерполяции по широте и долготе данных наблюдений, поставляемых с буёв ARGO, были проведены несколько экспериментов. Некоторые из них представлены в данном разделе.

Для начала приведём результаты работы алгоритма на данных измерений температуры воды, передаваемых с буёв ARGO, за январь 2008 года, на глубине 10 метров на акватории Индийского океана (рис. 6). В качестве следующего примера приведём распределение температуры воды, построенное методом интерполяции данных наблюдений по "горизонтальным переменным" за тот же период времени, но на глубине 75 метров (рис. 7). При сравнении рис. 6 и рис. 7 видно, что на глубине 75 метров температура воды океана имеет более низкий диапазон значений.

Для наглядности приведём ещё одно распределение температуры воды на акватории Индийского океана, но уже на глубине 200 метров от поверхности (рис. 8). Хорошо видно, что температуры воды существенно ниже на глубинах такого порядка, по сравнению с поверхностными температурами.

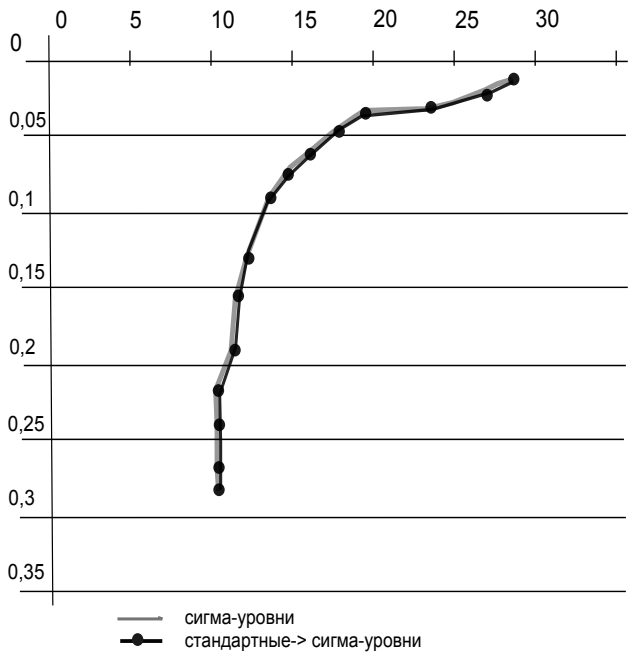


Рис. 4: Выходные данные программы для интерполяции данных на уровни в σ-координатах

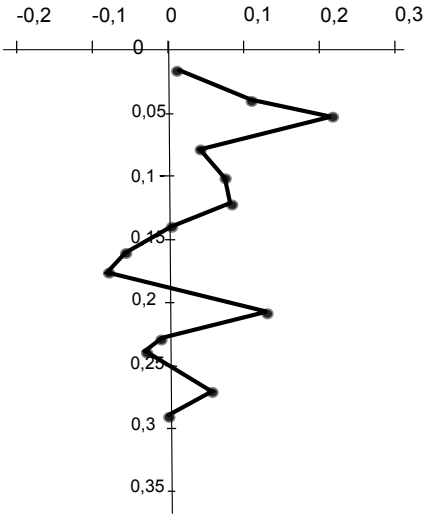


Рис. 5: Разница между данными, интерполированными непосредственно на уровни в σ-координатах и через промежуточный этап интерполяции на стандартные горизонты

Таким образом, разработанный Комплекс программ может быть использован для обработки, интерполяции и экстраполяции данных, передающихся с буйковых станций (глубинных). В настоящее время осуществляется работа по модернизации и развитию разработки Комплекса программ в целях получения полной картины поведения вод в океане.

В заключение, авторы выражают благодарность В.И. Агошкову за консультации и ценные замечания в процессе работы.

Настоящая работа выполнена при частичной поддержке РФФИ (проект № 07-01-00714)

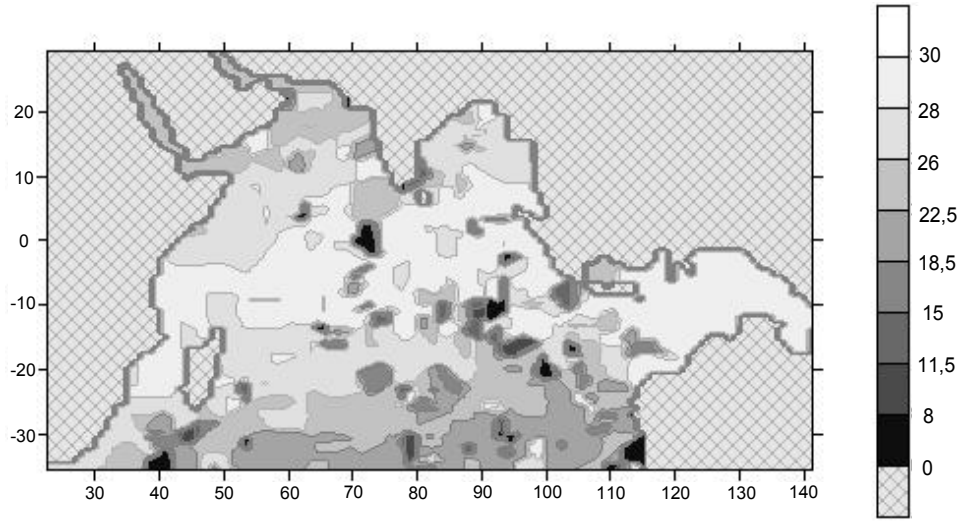


Рис. 6: Распределение температур на глубине 10 метров по данным с буёв ARGO за январь 2008 года на акватории Индийского океана

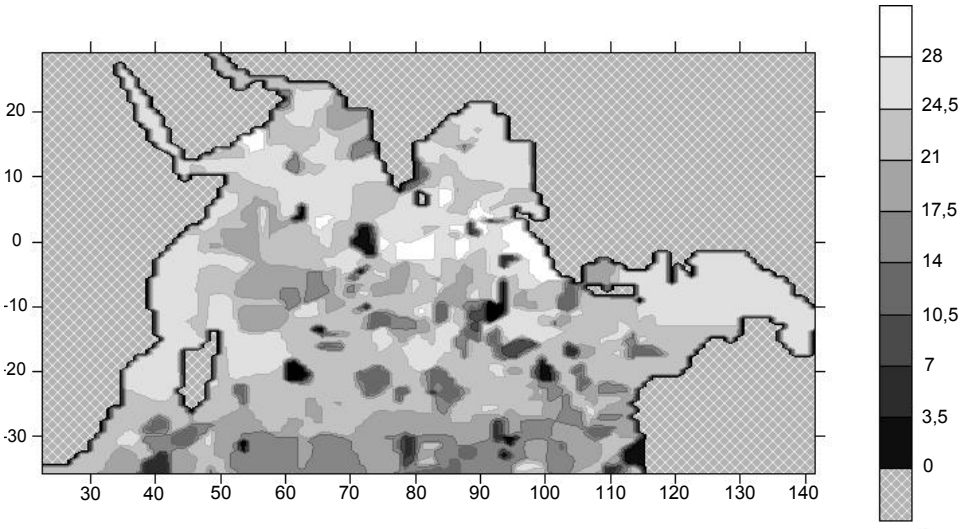


Рис. 7: Распределение температур на глубине 75 метров по данным с буёв ARGO за январь 2008 года на акватории Индийского океана

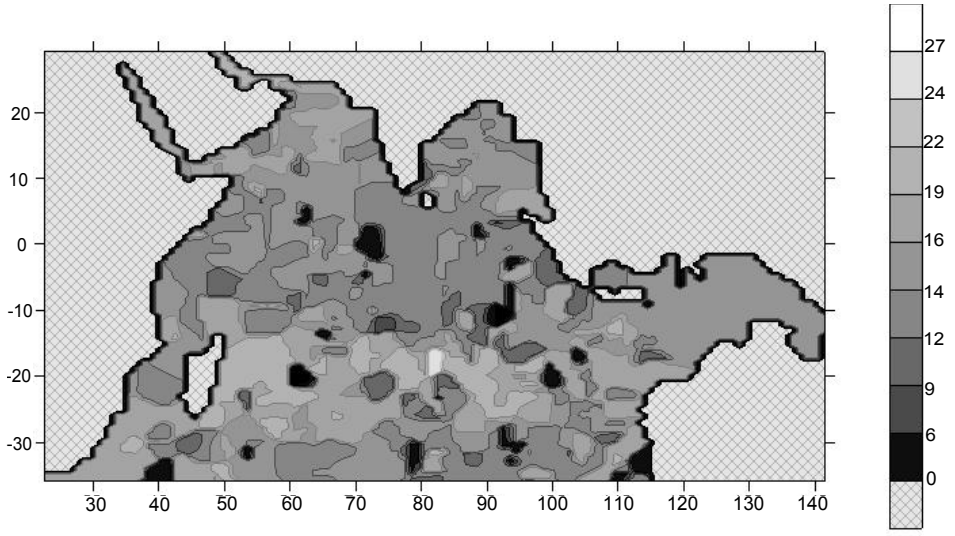


Рис. 8: Распределение температур на глубине 200 метров по данным с буёв ARGO за январь 2008 года на акватории Индийского океана

Список литературы

[1] Агошков В.И. *Методы оптимального управления и сопряженных уравнений в задачах математической физики.* - М: ИВМ РАН, 2003. - 258 с.

[2] Лебедев С.А., Агошков В.И. *Структура базы данных "Мировой океан – ИВМ РАН" Института вычислительной математики Российской академии наук.* //ИВМ РАН, Отчёт, 2006. - 43 с.

[3] Лебедев С.А., Агошков В.И. *База данных "Мировой океан - ИВМ РАН" Института вычислительной математики Российской академии наук.* //Сб: Материалы международной конференции '50-летие Международного геофизического года и Электронный геофизический год', 2007, ГЦ РАН, Москва, doi:10.2205/2007-IGY50conf.

[4] Рис У.Г. *Основы дистанционного зондирования.* //Пер. с англ. М.Б. Кауфмана и А.А. Кузьмичевой. – М: Техносфера, 2006. – 336 с. (Rees W.G. Physical Principles of Remote Sensing. 2nd Edition. – Cambridge University Press. 2001. – 372 pp.)

[5] Демидович Б.П., Марон И.А. *Основы вычислительной математики.* - М.: Наука, 1970. - 664 с.

[6] Казакевич Д.И. *Основы теории случайных функций в задачах гидрометеорологии.* - М.: Наука, 1989. - 230 с.

[7] Беляев В.И. *Обработка и теоретический анализ океанографических наблюдений.* - Киев: Наук. думка, 1973. - 296 с.

[8] Троян В.Н., Киселев Ю.В. *Статистические методы обработки и интерполяции геофизических данных.* - СПб: Изд-во СПбУ, 2003. - 576 с.

[9] Davis J.C. *Statistics and Data Analysis in Geology.* - John Wiley and Sons, New York, 1986. - 656 с.

[10] Stephens C., Antonov J.I., Boyer T.P., Conkright M.E., Locarnini R.A., O'Brien T.D. and Garcia H.E. *World Ocean Atlas 2001, Volume 1: Temperature.* /Ed. S. Levitus. - NOAA Atlas NESDIS 49. - U.S. Government Printing Office, Wash., D.C., 2002. - 176 pp.

[11] Agoshkov V., Botvinovsky E., Gusev A., Lebedev S., Parmuzin E. and Shutyaev V. *Variational data assimilation system INM-T1.* //Geophysical Research Abstracts, Vol. 10, EGU2008-A-08220, 2008, SRef-ID: 1607-7962/gra/EGU2008-A-08220, EGU General Assembly 2008.

[12] Агошков В.И., Ботвиновский Е.А., Гусев А.В., Кочуров А.Г., Лебедев С.А., Пармузин Е.И., Шутяев В.П. *Информационно-вычислительная система вариационной ассимиляции данных измерений ИВС-Т1.* //Сборник тезисов 6-ой всероссийской открытой конференции "Современные проблемы дистанционного зондирования Земли из космоса". 2008. Москва, ИКИ РАН. С.6.

[13] Агошков В.И., Пармузин Е.И., Шутяев В.П. *Численный алгоритм вариационной ассимиляции данных наблюдений о температуре поверхности океана.* //ЖВМ и МФ, 2008, Т. 48, № 8, С. 1371-1391.

[14] Агошков В.И., Лебедев С.А., Пармузин Е.И. *Численное решение задачи вариационной ассимиляции оперативных данных наблюдения температуры поверхности океана.* //Изв. РАН, Физика атмосферы и океана, 2009, № 1, С. 76-107.

[15] Бартеньев О.В. *Современный фортран.* - М.: ДИАЛОГ МИФИ, 2005. - 449 с.

[16] Захарова Н.Б. *Алгоритмы и программы интерполяции и экстраполяции в проблемах численного решения задач вариационной ассимиляции данных. База данных "Мировой океан - ИВМ РАН" и средства обеспечения работы с ней.* //ИВМ РАН, Отчёт, 2008. - 43 с.

РЕФЕРАТЫ

В. Ю. Антонов, А. П. Фокин, К. Н. Долгова. ВОССТАНОВЛЕНИЕ ТИПОВ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИИ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРОГРАММЫ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 6–16. Статья посвящена восстановлению типов данных с использованием информации времени выполнения программы

Библиография 6 раб.

А. И. Аристов. АСИМПТОТИКА ПРИ БОЛЬШИХ ВРЕМЕНАХ РЕШЕНИЯ ЗАДАЧИ КОШИ ДЛЯ УРАВНЕНИЯ СОБОЛЕВСКОГО ТИПА С АНАЛИТИЧЕСКОЙ НЕЛИНЕЙНОСТЬЮ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 17–22. Статья посвящена изучению асимптотики при больших временах обобщенного решения задачи Коши для уравнения соболевского типа. Исследуемое уравнение содержит нелинейность, имеющую вид степенного ряда. Показано, что главный член асимптотики определяется линейными членами уравнения, а нелинейность влияет на порядок остаточного члена.

Библиография 3 раб.

А. В. Беликов. ЕДИНСТВЕННОСТЬ ОБОБЩЕННЫХ РЕШЕНИЙ СМЕШАННЫХ ЗАДАЧ ДЛЯ ВОЛНОВОГО УРАВНЕНИЯ С НЕЛОКАЛЬНЫМИ ГРАНИЧНЫМИ УСЛОВИЯМИ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 23–35. Статья посвящена теоремам единственности обобщенных решений смешанных задач для волнового уравнения с нелокальными граничными условиями четырех типов с заданными либо начальными, либо финальными условиями.

Библиография 14 раб.

И. Е. Бронштейн, А. В. Столяров. БИБЛИОТЕЧНАЯ ПОДДЕРЖКА ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ МОДЕЛИ ЯЗЫКА РЕФАЛ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 36–46. В статье описывается подсистема библиотеки InteLib, моделирующая Рефал-вычислитель и структуры данных языка Рефал и дающая возможность использования вычислительной модели языка Рефал в рамках проектов на Си++

Библиография 11 раб.

С. Е. Бубнов. ФУНКЦИЯ ШЕННОНА ДЛИНЫ ПРОВЕРЯЮЩИХ ТЕСТОВ ФУНКЦИЙ, БЕСПОВТОРНЫХ В ЭЛЕМЕНТАРНОМ БАЗИСЕ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 47–57. Статья посвящена алгоритму построения проверяющих тестов для произвольных бесповторных функций в элементарном базисе.

Библиография 7 раб.

А. Б. Дайняк. О ЧИСЛЕ МАКСИМАЛЬНЫХ НЕЗАВИСИМЫХ МНОЖЕСТВ В ДЕРЕВЬЯХ ФИКСИРОВАННОГО ДИАМЕТРА // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 58–68. Устанавливается достижимая верхняя оценка числа максимальных независимых множеств в деревьях фиксированного диаметра. Приводится полное описание структуры экстремальных деревьев.

Библиография 6 раб.

Е. О. Деревенец, К. Н. Долгова. СТРУКТУРНЫЙ АНАЛИЗ В ЗАДАЧЕ ДЕКОМПИЛЯЦИИ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 69–80. Декомпиляция - одна из сложнейших задач обратной инженерии. Одной из подзадач декомпиляции является задача восстановления управляющих. Методы решения задачи восстановления управляющих конструкций рассматриваются в представляемой работе.

Также в статье подробно описан метод восстановления управляющих конструкций, реализованный в декомпиляторе TyDec, который разрабатывается Помимо этого в статье дается обзор метода, позволяющего восстанавливать работу с исключительными ситуациями на примере языка Си++.

Библиография 13 раб.

М. О. Гапонова, А. Ю. Корчагин, И. Г. Шевцова. ОБ АБСОЛЮТНЫХ КОНСТАНТАХ В РАВНОМЕРНОЙ ОЦЕНКЕ ТОЧНОСТИ НОРМАЛЬНОЙ АППРОКСИМАЦИИ ДЛЯ РАСПРЕДЕЛЕНИЙ, НЕ ИМЕЮЩИХ ТРЕТЬЕГО МОМЕНТА // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 81–89. Уточнены верхние оценки константы в неравенстве Берри–Эссеена для сумм независимых одинаково распределённых случайных величин с конечными абсолютными моментами порядка $2 + \delta$, $0 < \delta < 1$.

Библиография 30 раб.

В. Б. Ларионов. О ПОЛОЖЕНИИ САМОДВОЙСТВЕННЫХ k-ЗНАЧНЫХ ФУНКЦИЙ В РЕШЁТКЕ ЗАМКНУТЫХ КЛАССОВ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 90–104. В работе рассматривается одно из семейств замкнутых относительно операции суперпозиции классов функций в многозначных логиках — семейство самодвойственных классов. Изучается вопрос о строении надструктуры указанных классов.

Библиография 7 раб.

Д. В. Левшин. РЕЛЯЦИОННЫЕ ДАННЫЕ В ИНТЕГРИРОВАННЫХ СРЕДСТВАХ СЕМАНТИЧЕСКОГО WEB // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 105–110. Рассмотрены подходы к использованию РСУБД в средствах Семантического Web. Предложен метод представления данных из реляционных таблиц в RDF, который может использоваться в интегрированных средствах.

Библиография 21 раб.

А. С. Марков. О СКОРОСТИ СХОДИМОСТИ СПЕКТРАЛЬНЫХ РАЗЛОЖЕНИЙ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ ОПЕРАТОРОВ ВТОРОГО ПОРЯДКА. // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 111–127. В статье получены оценки скорости равносходимости с тригонометрическим рядом Фурье спектральных разложений широкого класса функций, имеющих особую асимптотику нормированных коэффициентов Фурье, на всем интервале. Показана зависимость скорости равносходимости от расстояния от компакта до границы интервала. Полученные оценки скорости равносходимости могут существенно зависеть от степени суммируемости s коэффициента при первой производной.

Библиография 10 раб.

А. Л. Назаров. РАЗДЕЛЕНИЕ СМЕСЕЙ ВЕРОЯТНОСТНЫХ РАСПРЕДЕЛЕНИЙ СЕТОЧНЫМ МЕТОДОМ МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ ПРИ ПОМОЩИ АЛГОРИТМА УСЛОВНОГО ГРАДИЕНТА // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 128–135. Статья посвящена применению метода условного градиента для поиска оценок максимального правдоподобия в задаче разделения смесей вероятностных распределений с помощью сеточного метода. Проводится анализ результатов применения алгоритма для решения задачи разделения смесей нормальных законов при использовании CPC-метода на примере индекса CAC40.

Библиография 5 раб.

А. А. Носков. МЕТОД ВЫДЕЛЕНИЯ В ТЕКСТЕ КОНСТРУКЦИЙ ПО ИХ ЛЕКСИКО-СИНТАКСИЧЕСКИМ ШАБЛОНАМ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 136–145. В статье описывается метод, позволяющий осуществлять автоматическое выделение конструкций в текстах на русском языке по их описанию в виде лексико-синтаксических шаблонов на языке LSPL

Библиография 3 раб.

A. I. Puchkova. OPTIMAL CONTROL IN THE SIMPLEST INVESTMENT ALLOCATION MODEL WITH INFINITE TIME HORIZON // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 146–157. The article is devoted to investigation of Dmitruk economic model at the infinite time horizon. The optimal solution is found for two special cases.

Библиография 7 раб.

О. В. Шестаков. О ПРОБЛЕМЕ ВОССТАНОВЛЕНИЯ КОЭФФИЦИЕНТА ПРЕЛОМЛЕНИЯ В ЗАДАЧАХ ДИФРАКЦИОННОЙ ТОМОГРАФИИ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 158–162. Основной задачей дифракционной томографии является восстановление коэффициента преломления изучаемого объекта. В работе рассматривается метод решения этой задачи в случае, когда коэффициент преломления предполагается нормированным и неотрицательным. Предлагается оценка скорости сходимости этого метода, позволяющая оценить степень неопределенности реконструкции в случае специального выбора направлений распространения волн.

Библиография 3 раб.

Е. А. Сытин. IrGene 1.0 — ИНТЕРФЕЙС И ПРОГРАММА ДЛЯ АНАЛИЗА ПОПУЛЯЦИОННО-ГЕНЕТИЧЕСКИХ ДАННЫХ В ИММУНОЛОГИИ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 163–167. Описаны возможности и интерфейс программы-оболочки IrGene 1.0, предназначенной для анализа различий генетических полиморфизмов популяций на основе таблиц сопряжённости. Программа отличается простым интерфейсом и представляет собой удобный инструмент исследований по проблеме “HLA и болезни”.

Библиография 13 раб.

Ю. В. Власенко, А. В. Столяров. ПРОГРАММИРУЕМЫЙ АГЕНТ ВЗАИМОДЕЙСТВИЯ ПО ПРОТОКОЛУ ПЕРЕДАЧИ ГИПЕРТЕКСТА // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 168–176. Статья посвящена вопросам реализации программы, способной выступать в роли клиента, сервера и прокси-сервера во взаимодействиях по протоколу HTTP и управляемой встроенным интерпретатором языка Лисп.

Библиография 8 раб.

Н. Б. Захарова, С. А. Лебедев. АЛГОРИТМЫ ИНТЕРПОЛЯЦИИ И ЭКСТРАПОЛЯЦИИ ОПЕРАТИВНЫХ ГЕОФИЗИЧЕСКИХ ДАННЫХ НАБЛЮДЕНИЙ // СБОРНИК СТАТЕЙ МОЛОДЫХ УЧЕНЫХ факультета ВМК МГУ, выпуск №6 (2009г.), стр. 177–188. В данной работе проводится обзор нескольких алгоритмов интерполяции и экстраполяции данных, описан класс оперативных геофизических данных наблюдений и приведены примеры некоторых численных экспериментов, осуществленных с помощью представленных алгоритмов применительно к описанным в работе оперативным данным наблюдений

Библиография 13 раб.
