НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В НАУКЕ И ОБРАЗОВАНИИ

Серия

“КОНСТРУИРОВАНИЕ И ОПТИМИЗАЦИЯ ПРОГРАММ”

Под редакцией
доктора физ.-мат. наук, профессора, чл.-корр. РАЕН
В. Н. Касьянова

Выпуски серии:

1. Смешанные вычисления и преобразование программ (1991)
2. Конструирование и оптимизация программ (1993)
3. Интеллектуализация и качество программного обеспечения (1994)
4. Проблемы конструирования эффективных и надежных программ (1995)
5. Оптимизирующая трансляция и конструирование программ (1997)
6. Проблемы систем информатики и программирования (1999)
7. Поддержка супервычислений и Интернет-ориентированные технологии (2001)
8. Касьянов В. Н., Мирзуитова И. Л. Slicing: срезы программ и их использование (2002)
9. Современные проблемы конструирования программ (2002)
10. ***Новые информационные технологии в науке и образовании***

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

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В НАУКЕ И ОБРАЗОВАНИИ

Под редакцией
проф. Виктора Николаевича Касьянова

Новосибирск 2003

УДК 519.68; 681.3.06
ББК З 22.183.49+ З 22.174.2

Новые информационные технологии в науке и образовании. — Новосибирск: Ин-т систем информатики им. А. П. Ершова СО РАН, 2003. — 303 с.

Является десятым в серии сборников, издаваемых Институтом систем информатики им. А.П.Ершова СО РАН по проблемам конструирования и оптимизации программ. Посвящен решению актуальных задач, связанных с применением новых информационных технологий в науке и образовании.

Сборник представляет интерес для системных программистов, а также студентов и аспирантов, специализирующихся в области системного и теоретического программирования.

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

NEW INFORMATIONAL TECHNOLOGIES IN SCIENCE AND EDUCATION

Edited by
prof. V. N. Kasyanov

Novosibirsk 2003

This volume is the tenth one in a series of books published in A.P. Ershov Institute of Informatics Systems on the problems of program construction and optimization. This volume is devoted to a decision of actual problems connected with using of new information technologies in science and education.

The volume is of interest for system programmers, students and post-graduates working in the field of system and theoretical programming.

ПРЕДИСЛОВИЕ РЕДАКТОРА

Десятый выпуск серии «Конструирование и оптимизация программ» посвящен решению актуальных задач, связанных с применением новых информационных технологий в науке и образовании.

Данный выпуск, как и предыдущие, базируется на результатах исследований, выполненных в лаборатории по конструированию и оптимизации программ ИСИ СО РАН совместно с Новосибирским государственным университетом (НГУ) при финансовой поддержке Российского фонда фундаментальных исследований (РФФИ), Российского гуманитарного научного фонда (РГНФ) и Минобразования РФ. Среди проектов три гранта РФФИ: «Методы и инструменты конструирования эффективных и надежных программ и систем» (01-01-00794), «Электронный толковый словарь по теории графов и ее применению в информатике и программировании, ориентированный на работу в среде Интернет» (00-07-90296), «Система быстрого прототипирования распараллеливающего транслятора» (02-07-90409), грант РГНФ 02-05-12010 «Виртуальный музей истории информатики в Сибири», грант Е02-1.0-42 «Граф-модели в программировании и визуальная обработка» конкурса Минобразования РФ по фундаментальным исследованиям в области естественных и точных наук, а также грант УР.04.01.023 «Графы в программировании: обработка, визуализация и применение» научной программы Минобразования РФ «Фундаментальные исследования высшей школы в области естественных и гуманитарных наук. Университеты России».

Одним из основных результатов указанных работ стала книга В.Н. Касьянова и В.А. Евстигнеева «Графы в программировании: обработка, визуализация и применение» (СПб.: БХВ-Петербург, 2003, 1104 С.), выход в свет которой совпал по времени с завершением комплектования данного сборника. Впервые издана книга, которая содержит систематическое и полное изложение фундаментальных основ современных компьютерных технологий, связанных с применением теории графов. Даны основные модели, методы и алгоритмы прикладной теории графов. Рассмотрены задачи рисования графов и визуальной обработки графовых моделей. Подробно описаны такие основные области приложения, как хранение и поиск информации, трансляция и оптимизация программ, анализ, преобразование и распараллеливание программ, параллельная и распределенная обработка информации. В книге используется высокоуровневое описание алгоритмов, позволяющее понять алгоритм на содержательном уровне, оценить пригодность его для решения конкретной задачи и осуществить модификацию алгоритма, не снижая степень математической достоверности окончательного варианта программы.

Книга состоит из 12 глав и 2 приложений, образующих три части. Часть I посвящена обработке и визуализации графов. Основная задача этой части, состоящей из пяти глав, — описать базовые модели и алгоритмы, связанные с применением теории графов в программировании, включая вопросы визуализации и рисования графов. Изложение материала привязано к трём основным типам графов: деревья, дэги, или бесконтурные графы, и сводимые, или регуляризуемые, графы. Отдельно рассматриваются основные теретико-графовые понятия и методы рисования и визуальной обработки графовых моделей. В части II, состоящей из семи глав, сосредоточен основной материал по применению графов и граф-моделей в программировании и информатике. В ней подробно рассмотрены использования графовых моделей в таких основных областях приложения, как хранение и поиск информации, трансляция и оптимизация программ, анализ, преобразование и распараллеливание программ, параллельная и распределенная обработка информации. Часть III содержит справочный материал и состоит из двух приложений. Прил. 1 посвящено «переборным» (NP-полным) задачам и содержит достаточно полный список известных NP-полных задач из различных областей программирования и информатики. В прил. 2 приведены характеристики размещений графов, а также размер области размещения, оценки величины углов, число сгибов и пр.

Современное состояние информационного общества нельзя представить себе без применения теоретико-графовых методов и алгоритмов. Теория графов из академической дисциплины все более превращается в средство, владение которым становится решающим для успешного применения компьютеров во многих прикладных областях. Академик Андрей Петрович Ершов, основатель сибирской школы информатики и программирования, называл графы основной конструкцией для программиста и говорил, что «графы обладают огромной, неисчерпаемой изобразительной силой, соразмерной масштабу задачи программирования».

Причем совершенно ясно, что, несмотря на наличие обширной специальной литературы по решению задач на графах, широкое применение в практике программирования полученных математических результатов затруднено в силу отсутствия систематического их описания, ориентированного на программиста. Изданная книга призвана, если не решить, то во многом упростить решение данной задачи.

Проф. В.Н. Касьянов

Т.В. Батура, О.Н. Еркаева, Ф.А. Мурзин

К ВОПРОСУ ОБ АНАЛИЗЕ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ*

ВВЕДЕНИЕ

В работе рассмотрены различные понятия, конструкции и методы, применяемые в математической и классической лингвистике для анализа лексики и текстов на естественном языке.

Первые три параграфа посвящены подходу Мельчука и Аппресяна [1, 2]. Дается формализованное описание таких понятий, как лексические функции, валентности, толково-комбинаторный словарь.

Четвертый параграф посвящен теоретико-множественным моделям Маркуса [3]. Подход Маркуса применим практически к любым языкам и позволяет на основе чисто структурного анализа определить категории рода и падежа в данном языке. Можно предположить, что посредством различных модификаций подхода Маркуса можно определить самые различные категории языка.

Далее рассматриваются словообразовательные конструкции русского языка и минимальные схемы предложений [4, 5]. Они могут быть полезны не только для морфологического и синтаксического, но и для семантического анализа.

1. ЛЕКСИЧЕСКИЕ ФУНКЦИИ

Лексические функции — понятие, известное в лингвистике. Наша цель — представить лексические функции на синтаксическом уровне в виде предикатов. Позднее предполагается рассмотреть различные модели соответствующих сигнатур, и можно будет говорить об истинности данных предикатов в моделях.

Обозначим через L совокупность правильно построенных слов какого-либо естественного языка. Имеется в виду, что всевозможные словоформы,

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

возникающие при склонениях существительных, спряжениях глаголов и др., также включены в L. Пусть x, y, z — слова или словосочетания этого языка L.

Перечень простых стандартных лексических «функций» (у нас они будут представлены в виде предикатов) приведен ниже.

1. $Syn(x, y)$, x, y — синонимы.
   $Syn_{\supset}(x, y)$, x, y — синонимы, но x более широкий по смыслу.
2. $Conv(x, y)$, x, y — конверсивы.
3. $Anti(x, y)$, x, y — антонимы.
4. $Der(x, y)$, y — синтаксический дериват x, т.е. y совпадает с x по смыслу, но принадлежит к другой части речи:
   $S_0(x, y)$, y — существительное, производное от x (x — не существительное).
   $A_0(x, y)$, y — прилагательное, производное от x (x — не прилагательное).
   $Adv_0(x, y)$, y — наречие, образованное от x (x — не наречие).
   $V_0(x, y)$, y — глагол, образованный от x (x — не глагол).
   Иначе говоря,
   $\forall x \forall y(Der \quad (x, y) \leftrightarrow S_0(x, y) \vee A_0(x, y) \vee Adv_0(x, y) \vee V_0(x, y))$.
5. $Gener(x, y)$, y — название понятия, родового по отношению к понятию, обозначенному x (x= «клубника», y = «ягода»). Этот предикат зависит от лексической сочетаемости слов в данном языке: если x и m — совпадающие по смыслу слова двух разных языков, то для $Gener(x, y)$ и $Gener(m, n)$ соответственно *y* и *n* могут не совпадать по смыслу.

Ситуация — определенное лексическое отражение (в данном языке) некоторой части действительности. Ситуации, обозначаемые отдельными лексическими единицами естественных языков (лексемами), имеют, как правило, от одного до четырех «смысловых» компонентов, или **семантических актантов,** обозначаемых заглавными латинскими буквами A, B, C, D. В то же время каждой такой лексеме сопоставляются **глубинно-синтаксические актанты** — ее зависимые, соответствующие подлежащему и сильным дополнениям (в случае, если данная лексема реализуется гла-

голом-сказуемым). Глубинно-синтаксические актанты нумеруются арабскими цифрами: 1, 2, 3, 4.

6. $S_i(x,y),\ i=1,...,4$, y — типовое название i-того актанта для x.
7. $S_c(x,y)$, y — сирконстанта, т.е. типовое название второстепенной компоненты данной ситуации x:
   $S_{loc}(x,y)$, y — типовое название места осуществления данной ситуации x (= «то, где…», например, x = «битва», y = «поле (битвы)»).
   $S_{instr}(x,y)$, y — типовое название инструмента, используемого в данной ситуации x (= «то, чем/посредством чего…», x = «борьба», y = «орудие (борьбы)»).
   $S_{\mathrm{mod}}(x,y)$, y — типовое название способа (манеры, характера) осуществления данной ситуации x (= «то, как…», x = «жизнь», y = «образ (жизни)»).
   $S_{res}(x,y)$, y — типовое название результата данной ситуации (= «то, что получается», x = «копировать», y = «копия»).
   Иначе говоря,
   $\forall x \forall y (S_c(x,y) \leftrightarrow S_{loc}(x,y) \vee S_{instr}(x,y) \vee S_{\mathrm{mod}}(x,y) \vee S_{res}(x,y) \vee ...)$.
8. Соотносительные предикаты $Sign\ (x,y)$, y — типовое название одной «штуки», одного «кванта» некоторого x; $Mult\ (x,y)$, y — типовое название совокупности, множества.
9. $Figur\ (x,y)$, y — метафора для x (x = «сон», y = «объятия (сна)»).
10. $Centr\ (x,y)$, y — типовое обозначение «центральной» части предмета или процесса.
11. $A_i(x,y),\ i=1,...,4$, y — типовое определение i-го актанта по его реальной роли («такой, который…», «такой, которого…»).
12. $Able_i(x,y),\ i=1,...,4$, y — типовое определение i-го актанта по его потенциальной роли в ситуации («такой, который может...», «такой, которого можно...»).
13. $Magn_0(x,y)$ и $Magn_i(x,y),\ i=1,...,4$, y обозначает «высокую степень», «интенсивность» самой ситуации x ($Magn_0$) или ее i-го актанта ($Magn_i$).
14. $Ver\ (x,y)$, y — «правильный», «соответствующий назначению», «какой следует» применительно к x.

15. $Bon\ \ (x,y)$, y — «хороший» применительно к x.
16. $Adv_{ix}(z,y),\ i=1,...,4;\ x=A,B,C,D$, y — имя ситуации в роли определения при глаголе, называющем другую ситуацию:
    $Adv_{iA}(z,y),\ i=1,...,4$, y — слово, образованное от z, которое, заменяя z в тексте, требует превратить в вершину (вместо z) первый актант этого z (x = «сопровождать», y = «вместе с»).
    $Adv_{iB}(z,y),\ i=1,...,4$, y требует становиться вершиной второй актант z и т.д. (x=«ошибаться», y = «ошибочно»).
17. $Loc(x,y)$, y — предлог типовой локализации (пространственной, временной или абстрактной):
    $Loc_{in}(x,y)$, y — «статическая» локализация (если x = «Москва», то y = «в Москве»).
    $Loc_{ad}(x,y)$, y — предлог направления (если x = «Москва», то y = «в Москву»).
    $Loc_{ab}(x,y)$, y — предлог удаления (если x = «Москва», то y = «из Москвы»).
    Иначе говоря, $\forall x\forall y(Loc(x,y)\leftrightarrow Loc_{in}(x,y)\vee Loc_{ad}(x,y)\vee Loc_{ab}(x,y))$.
    Иногда $Loc(x,y)$ не удается определить однозначно (x = «снег», y = «на» и y = «в»).
18. $Copul\ \ (x,y)$, y — глагол-связка «быть», «являться» (x = «он нападал», y = «он совершал нападения»).
19. $Oper_1(x,y),\ Oper_2(x,y)$, y — глагол, связывающий название первого (соответственно второго) актанта в роли подлежащего названием ситуации в роли первого дополнения (если x = «поддержка», то y = «оказывать» для $Oper_1(x,y)$, а y = «находить» или «встречать» для $Oper_2(x,y)$).
20. $Func_0(x,y),\ Func_1(x,y),\ Func_2(x,y)$, y — глагол, описывающий ситуацию; x — названия актантов, если они есть, являющихся подлежащим или дополнениями (x = «дождь», y = «идти»).
21. $Labor_{12}(x,y)$, y — глагол, связывающий название первого актанта в роли подлежащего, с названием второго актанта в роли первого дополнения и с названием ситуации в роли второго дополнения (x = «орден», y = «награждать»; x = «наказание», y = «подвергать»).

22. $Caus_{ij}(x, y)$, y — «делать так, чтобы…», «каузировать» (действие актантов). В случае $Caus\ (x, y)$ (без актантных индексов) x — название неучастника ситуации (x = «преступление», y = «толкать»). Отдельно выступает только при глаголах, в остальных случаях входит в состав сложных параметров.
23. $Perm_{ij}(x, y) = не\ Caus_{ij}(x, y)\ не$ (x = «преступление», y = «не препятствовать»).
24. $Liqu_{ij}(x, y) = неCaus_{ij}(x, y)$ (x = «преступление», y = «мешать»).
25. $Incep\ (x, y)$, y — «начинать». Свойства те же, что и у $Caus_{ij}(x, y)$.
26. $Cont\ (x, y) = не\ Incep(x, y)\ не$, т.е. y — «продолжать» = «не переставать».
27. $Fin\ (x, y) = Incep\ (x, y)\ не$, т.е. y — «переставать» = «начинать не».
28. $Perf\ (x, y)$ («перфектив»), y несет завершенность действия, достижение им своего естественного предела. Отдельного самостоятельного выражения $Perf\ (x, y)$ в русском языке не имеет; как правило, этот предикат выдает истинное значение, если y имеет форму совершенного вида.
29. $Result\ (x, y)$ («результатив»), т.е. y — «состояние в результате…»; используется для форм несовершенного вида (x = «ложиться», y = «лечь» для $Perf\ (x, y)$, y = «лежать» для $Result\ (x, y)$).
30. $Fact^{j}(x, y)$, y — «реализоваться», «выполниться». Верхний индекс (римские цифры) представляет, если это надо, степень осуществления подразумеваемого требования, причем меньший индекс присваивается более низкой степени (если x = «капкан» и j = I, то y=«срабатывать»; если j = II, то y = «поймать»).
31. $Real^{j}_{1,2}(x, y)$, y — «реализовать», «выполнить» «требование», содержащееся в x. Индекс j имеет то же значение, что и выше — степень выполнения; нижний индекс обозначает глубинно-синтаксический актант, выполняющий требование (x = «долг (денежный)», y = «признавать» для $Real^{I}_{1,2}(x, y)$, y = «погашать» для $Real^{j}_{1,2}(x, y)$).
32. $Prepar^{j}(x, y)$, y — «приводить в полную готовность к» употреблению, функционированию и т. п. Индекс j, как и в предикатах $Fact^{j}(x, y)$,

$Real^{j}_{1,2}(x,y)$, показывает степень готовности (x = «ружье», y = «заряжать» для $Prepar^{I}(x,y)$, y = «взводить курок» для $Prepar^{j}(x,y)$).

33. $Degrad(x,y)$, y — «портиться», «выходить из строя» (= $Incep\,\mathrm{Pr}edAntiVer(x,y)$: «начинать быть не таким, каким следует» или $IncepPejor(x,y)$: «начинать быть хуже»).
34. $Imper(x,y)$ («императив»), y — значение «повеления»; для глаголов y — повелительное наклонение (x = «уходить», y = «брысь» или y = «кышь» (кошке); x = «вставать II», y=«подъем!» (в армии)).
35. $Son(x,y)$, y — название типового звучания x (x = «корова», y = «мычать» или y = «му-у-у»).
36. $Destr(x,y)$, y — типовое название «агрессивного» действия (x = «оса», y = «жалит»).
37. $Cap(x,y)$, y — «начальник» (x = «факультет», y = «декан»).
38. $Equip(x,y)$, y — «личный состав» (x = «население», y = «государства»).
39. $Doc(x,y)$, y — «документ»:

    $Doc_{res}(x,y)$, y — «документ», являющийся результатом («воплощающий в себе»; x = «отчитываться», y = «отчет»).

    $Doc_{perm}(x,y)$, y — «документ на право…» (x = «поезд», y = «(проездной) билет» для $Doc_{perm}Oper_{2}(x,y)$).

    $Doc_{cert}(x,y)$, y — «документ, удостоверяющий…» (x = «высшее образование», y = «диплом»).

    Иначе говоря,
40. $Attr(x,y)$, y — типовая метонимия для x (x = «школьник», y = «парта»; x = «офицер», y = «погоны»).

Помимо перечисленных выше простых лексических предикатов для описания лексической сочетаемости могут использоваться и их комбинации — составные предикаты:

$Anti\,\mathrm{Re}al_{2}(x,y)$: проваливать экзамен /проваливаться на экзамене.

$IncepOper_{1}(x,y)$: приобретать популярность, впадать в отчаяние.

$IncepOper_{2}(x,y)$: поступать в продажу, попадать под обстрел.

$CausOper_{2}(x,y)$: ставить под контроль, пускать в обращение.

В общем случае лексическая функция определяется не для всех слов и словосочетаний.

Во-первых, некоторые лексические функции определены лишь для той или иной части речи; так, *Oper* , *Func* и *Labor* мыслимы лишь для существительных, а *Incep* , *Cont* и *Fin* — лишь для глаголов. В случае, когда лексические функции представляются в виде предикатов, то это не вызывает затруднений. В данном случае соответствующие предикаты ложны.

Во-вторых, та или иная функция может определяться только для слов определенной семантики. Например: *Magn* — для слов, смысл которых допускает градацию «больше — меньше»; *Cap* и *Equip* — для слов, смысл которых предполагает наличие «начальника» и «персонала», т. е. для названий учреждений и организаций в самом широком понимании. Далее, *Conv* — для слов, называющих отношения с двумя и более местами; *Real* — для слов, в смысл которых входит компонент «требовать» («нужно») и т. д. Как уже отмечалось, *Oper* , *Func* и *Labor* определены только для названий ситуаций.

Следует иметь в виду, что и при вполне подходящем (по своим синтаксическим и семантическим свойствам) аргументе лексическая функция может не иметь значения (в данном языке). Например, синонимы, в принципе, возможны для любых слов, а имеются только у некоторых: при $Oper_1$ *(визит)* = офиц. *наносить* [~ Sдат], *являться (приходить)* [*с ~ом к S*дат], $Oper_2$ для слова *визит* в современном русском языке отсутствует (по смыслу должно было бы быть что-то вроде **получать, *принимать,* подвергаться)*.

2. ВАЛЕНТНОСТИ

Валентностями обладают слова, которые являются предикатами, т. е. те, которые задают ситуацию. Это все глаголы, некоторые существительные (отглагольные), прилагательные (обозначающие сравнение: больше, меньше, выше, ниже), некоторые предлоги и наречия.

Валентности слова бывают синтаксические и семантические. Семантические валентности определяются лексическим анализом ситуации, задаваемой этим словом. Приведём пример со словом *аренда* или *арендовать*. A *арендует* C значит, в первом приближении, что за какое-то вознаграждение D лицо A приобретает у другого лица B право на эксплуатацию недви-

жимой собственности С в течении времени Т. Следовательно, существенными для ситуации аренды являются следующие «участники» или семантические актанты: субъект аренды (тот, кто арендует), первый объект аренды (то, что арендуют), контрагент (тот, у кого арендуют), второй объект (плата) и срок.

Эти актанты необходимы, так как устранение какого-либо из них изменяет смысл ситуации. Например если убрать срок, то ситуация аренды трансформируется в ситуацию купли-продажи. С другой стороны, эти актанты достаточны, поскольку в ситуации аренды не требуется указание того, по какой причине, где, когда и с какой целью она осуществлялась. Хотя соответствующие словоформы грамматически присоединимы к глаголу *арендовать*.

В итоге, эта ситуация имеет 5 валентностей и формально записывается в виде предиката $P(x_1, x_2, x_3, x_4, x_5)$, где x_1 — «кто», x_2 — «что», x_3 — «у кого», x_4 — «цена», x_5 — «срок».

В предложении могут быть определены актанты не для всех семантических валентностей, некоторые могут просто не упоминаться или вообще не иметь синтаксического выражения.

Синтаксические валентности — это те, которые представлены в тексте. Они определяются присоединяемыми к слову подлежащими и дополнениями и зависят от контекста.

Например, глагол *промахнуться* имеет 4 семантические валентности: кто (деятель), во что/по чему (мишень), из чего (оружие — факультативно) и чем (орган, средство). Но в большинстве контекстов синтаксически выражается лишь первая валентность. Например, нельзя сказать *«Он промахнулся в окно бутылкой»*.

Возможны случаи, когда синтаксических валентностей у слова больше, чем семантических.

Для того чтобы не связывать с каждым глаголом (и другими словами) отдельный предикат, будем рассматривать предикат, размерность которого больше на 1: $P^{val}(y, x_1, x_2, \ldots, x_n)$, при этом y будет само слово, а $x_1, x_2, \ldots, x_n$ — его валентности. Чтобы отличать синтаксические и семантические актанты, можно использовать мультииндексы с целью указать, какие актанты заданы в тексте. Запись $P^{val}_{i_1 i_2 \ldots i_k}(y, x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ означает, что заданы актанты $i_1, i_2, \ldots, i_k$. В частности если заданы все актанты, то получаем $P^{val}_{1\ldots n}(y, x_1, x_2, \ldots, x_n)$. Некоторые варианты (наборов мультииндексов) могут

быть недопустимы в языке. Если набор $i_1, i_2, \ldots, i_k$ допустим, то имеет место импликация:

$$\forall y \forall x_1 \ldots \forall x_n (P^{val}_{1\ldots n}(y, x_1, x_2, \ldots, x_n) \to P^{val}_{i_1 i_2 \ldots i_k}(y, x_{i_1}, x_{i_2}, \ldots, x_{i_k})) .$$

Более того, если имеется два набора допустимых мультииндексов $\langle i_1, i_2, \ldots, i_k \rangle$ и $\langle i_1', i_2', \ldots, i_s' \rangle$ таких, что $\{i_1, i_2, \ldots, i_k\} \supseteq \{i_1', i_2', \ldots, i_s'\}$, то имеет место аналогичная импликация:

$$\forall y \forall x_{i_1} \ldots \forall x_{i_k} \forall x_{i_1'} \ldots \forall x_{i_s'} (P^{val}_{i_1 \ldots i_k}(y, x_{i_1}, x_{i_2}, \ldots, x_{i_k}) \to P^{val}_{i_1' i_2' \ldots i_s'}(y, x_{i_1'}, x_{i_2'}, \ldots, x_{i_s'})) .$$

3. ТОЛКОВО-КОМБИНАТОРНЫЙ СЛОВАРЬ

Можно считать, что язык — это очень большая модель, в которой определены лексические предикаты, действующие соответствующим, описанным выше образом.

Статья толково-комбинаторного словаря несет информацию о валентностях конкретного слова, верную не только в ее рамках, но и в рамках всего языка в целом. Валентности соответствует предикат $P^{val}(c_x, \bar{y})$, где $\bar{y} = y_1, \ldots, y_n$ — семантические актанты слова c_x, n — валентность слова c_x. Например, в предложении *Петя читает книгу* будет $c_x = "читать"$, $n = 2$: $y_1 = "Петя"$, $y_2 = "книга"$, т.е. условно можно написать $P^{val}(c_x, y_1, y_2) = 1$.

Набор статей в толково-комбинаторном словаре можно считать некоторой подмоделью исходной модели, являющейся языком. Лексические предикаты, определенные теперь на более узком множестве, будут действовать аналогично.

Пусть Φ — множество правильно построенных фраз языка L, и $\varphi \in \Phi$ – фраза из этого множества, $c_x \prec \varphi$ — слово c_x входит во фразу φ, причем $c_x \in$ L, а c_x — существительное или прилагательное. Обозначим Predicate — множество предикатов, определенных на L. Одним из элементов этого множества является введенный ранее предикат валентности $P^{val}(c_x, y_1, \ldots, y_n)$.

Аналогично, можно предполагать, что имеются другие предикаты:

- предикат рода слова $Case(c_x, \overline{Cas})$, где
  $\overline{Cas} \in \{cas_1 = "жен."; \ cas_2 = "муж."; \ cas_3 = "ср."\}$;
- предикат предлога $Preposition(c_x, \overline{Prep})$, где $\overline{Prep} \in \{pr_1, ..., pr_k\}$ — множество предлогов, сочетаемых с данным словом;
- предикат падежа $Gender(c_x, \overline{G})$, где $\overline{G}$ — падеж слова c_x. Для разных языков число падежей различно. Например, для русского языка имеется шесть падежей:
  $g_1 = "им.п."; \ g_2 = "род.п."; g_3 = "дат.п."; g_4 = "вин.п."$;
  $g_5 = "твор.п."; g_6 = "предл.п."$,
  для немецкого — четыре падежа:
  $g_1 = "N"; g_2 = "G"; g_3 = "D"; g_4 = "A"$.

Словарная статья толково-комбинаторного словаря имеет следующий вид. Она содержит основное слово, лексические предикаты, связанные с ним, и информацию о валентности данного слова.

Информация о валентности включает в себя число, указывающее число актантов, и для каждого актанта — указание, в каких падежах и с какими предлогами используются слова, соответствующие данному актанту. В отдельных случаях может быть указан также род слова.

Сказанное выше может быть представлено посредством набора предикатов вида

$$P(x_i, \overline{Cas}, \overline{Prep}, \overline{G}) = Case(x_i, \overline{Cas}) \,\&\, Preposition(x_i, \overline{Prep}) \,\&\, Gender(x_i, \overline{G}),$$

где x_i — свободная переменная, соответствующая i-му актанту.

4. ЧАСТИ РЕЧИ И СИНТАКСИЧЕСКИЕ ТИПЫ, СТРУКТУРНЫЕ МЕТОДЫ МАРКУСА

4.1. Основные определения и обозначения

Назовем конечное множество слов *словарем* Γ. Рассмотрим свободную полугруппу T на Γ, а именно, множество всех конечных последовательностей слов с определенной на нем ассоциативной и некоммутативной бинарной операцией конкатенации (сцепления). Последовательность слов мы бу-

дем называть также последовательностью (цепочкой) над Γ. *Нулевая последовательность*, обозначим ее θ, — это такая последовательность, что $\theta \quad x = x\theta = x$ для каждой последовательности *x*. Если это специально не оговорено, θ не принадлежит Γ.

Подмножество $\Phi \subseteq T$ назовем *языком* над Γ. Полугруппа T в целом будет называться в этом случае *полным* или *универсальным языком* над Γ.

Тройку $\{\Gamma, P, \Phi\}$, где Γ — конечный словарь, P — разбиение Γ, а Φ — подмножество свободной полугруппы над Γ, мы назовем *языком с парадигматической структурой* или просто *языком*.

При машинном переводе требуется рассматривать языки с более сложной структурой. При этом в качестве языка должна рассматриваться система $\{\Gamma, P, \Phi, K, \varphi\}$, где Γ, P и Φ — определенные выше объекты, K — класс подмножеств Γ, называемых грамматическими категориями (например, множество слов в именительном падеже и множество слов в прошедшем времени), а φ — функция, сопоставляющая каждому слову *x* пересечение всех грамматических категорий, содержащих *x*.

Рассмотрим наиболее простое понятие языка, который в этом случае задается как пара $\{\Gamma, \Phi\}$. Последовательности, принадлежащие Φ, мы назовем *отмеченными последовательностями*.

Наиболее важным разбиением Γ, которое встречается в лингвистике, является так называемое разбиение на *дистрибутивные классы*. Оно определяется следующим образом. Два слова *a* и *b* принадлежат одному дистрибутивному классу, если для каждой пары последовательностей *x, y* из $xay \in \Phi$ следует $xby \in \Phi$, и обратно.

Введем понятие *контекста*. Контекст может быть определен как упорядоченная пара последовательностей над Γ. Мы будем обозначать контекст $\langle x, y \rangle$, где $x \in T$ и $y \in T$. Слово *a допустимо* в контексте $\langle x, y \rangle$, если последовательность *xay* принадлежит Φ. Обозначим через $\mathbf{J}(a)$ множество всех контекстов, в которых допустимо *a*. Отсюда непосредственно следует, что два слова *a* и *b* принадлежат одному дистрибутивному классу тогда и только тогда, когда $\mathbf{J}(a) = \mathbf{J}(b)$, т.е. тогда и только тогда, когда *a* и *b* допустимы в одних и тех же контекстах.

Легко видеть, что два различных дистрибутивных класса не пересекаются; таким образом, эти классы определяют разбиение Γ, которое мы назовем *дистрибутивным разбиением* Γ. Дистрибутивный класс можно назы-

вать также *семейством*. Семейство, содержащее *a*, мы будем обозначать $S(a)$. Мы будем употреблять эти два термина как эквивалентные.

Более сложным является понятие языка как тройки $\{\Gamma, P, \Phi\}$, где P — разбиение Γ, отличное от разбиения на дистрибутивные классы. Если P — разбиение Γ, то каждое подмножество P мы будем *называть клеткой из P* или *P-клеткой*. Если определено разбиение P, то мы можем написать: $\Gamma = \bigcup_{i=1}^{n} P_i$, где P_i обозначает клетку из P, а число клеток равно n. Поскольку подмножества P_i не пересекаются, каждое слово принадлежит одной и только одной клетке. Обозначим через $P(a)$ клетку из P, содержащую слово a. Из сказанного следует, что для двух различных слов a и b выполняется либо $P(a) = P(b)$, либо $P(a) \cap P(b) = 0$.

Множество $P(a)$ обычно интерпретируется в естественных языках как множество флективных форм слова a. Такая интерпретация требует введения так называемого *единичного разбиения* Γ, в котором каждая клетка состоит из одного слова.

Рассмотрим два разбиения P и Q словаря Γ. Будем говорить, что разбиение *P мельче* разбиения Q, если $P(a) \subseteq Q(a)$ для каждого $a \in \Gamma$.

Единичное разбиение является самым мелким разбиением Γ, а любое разбиение Γ мельче несобственного разбиения. Если мы интерпретируем $P(a)$ как множество всех флективных форм a, то разбиение P окажется мельче, чем разбиение Γ на части речи.

Если $x_1 x_2 \ldots x_n$ — последовательность над Γ, то последовательность $P(x_1)P(x_2)\ldots P(x_n)$ называется *P-структурой* последовательности $x_1 x_2 \ldots x_n$. *P-структура* является *отмеченной*, если последовательность $x_1 x_2 \ldots x_n$ может быть выбрана так, что она принадлежит Φ. Другими словами, *P*-структура $P_1 P_2 \ldots P_s$ отмечена, если существует отмеченная последовательность $x_1 x_2 \ldots x_n$ такая, что $P_i = P(x_i)$ при $1 \le i \le s$.

Далее будем говорить, что *P регулярно мельче Q*, если *P* мельче *Q*, и для каждой тройки слов *x, y, z* выполнение включений $P(x) \subseteq Q(z)$ и $P(y) \subseteq Q(z)$ влечет *P*-эквивалентность $P(x) \leftrightarrow P(y)$.

В качестве простейшего примера регулярно более мелкого разбиения возьмем единичное разбиение *E*. Это разбиение регулярно мельче, чем разбиение *S* на дистрибутивные классы. Для каждого разбиения *P* над Γ рас-

смотрим разбиение P', клетки которого определены следующим образом: $P'(x) = \bigcup_{P(x) \leftrightarrow P(y)} P(y)$ (для каждого $x \in \Gamma$), где объединение берется по всем словам y, для которых $P(y) \leftrightarrow P(x)$.

По своему определению разбиение P' таково, что P регулярно мельче P'. Разбиение P' назовем *производным* от разбиения P. Следует отметить, что разбиение S на дистрибутивные классы является производным от единичного разбиения E: $S = E'$. В языке $\{\Gamma, P, \Phi\}$ производное P' определено однозначно. Но может существовать несколько языков вида $\{\Gamma, P, \Phi\}$, для которых производное будет одинаковым.

Рассмотрим язык $\{\Gamma, P, \Phi\}$. *Цепью между словами* a и b назовем конечную последовательность $x_1, x_2, ..., x_i, x_{i+1}, ..., x_n$ такую, что $x_1 = a$, $x_n = b$ и $x_i \in S(x_{i+1}) \cup P(x_{i+1})$ для $1 \le i \le n-1$.

Число n определяет длину цепи. Будем считать, что для каждого слова a существует цепь длиной 1, соединяющая a с самим собой.

Обозначим $R(a)$ множество таких слов b, для каждого из которых существует цепь, соединяющая a и b. Как легко видеть, (1) $a \in R(a)$; (2) если $b \in R(a)$, то $a \in R(b)$; (3) если $b \in R(a)$ и $c \in R(b)$, то $c \in R(a)$. Таким образом, множества $R(a)$ при $a \in \Gamma$ определяют разбиение R словаря Γ. Такое разбиение мы назовем *разбиением из смешанных клеток*. Для каждого $a \in \Gamma$ выполняется $S(a) \subseteq R(a)$ и $P(a) \subseteq R(a)$. Отсюда видно, что разбиения P и S мельче R. Объединение P и S составляет R.

Обозначим через $K(a)$ множество слов b таких, что выполняется по крайней мере одно из двух следующих условий: $P(a) \cap S(b) \neq 0$ (1); $P(b) \cap S(a) \neq 0$ (2). Множество $K(a)$ по определению является классом a. Поскольку $a \in P(a) \cap S(a)$, то следовательно, $a \in K(a)$ для каждого $a \in \Gamma$.

4.2. Типология языков

Язык $\{\Gamma, P, \Phi\}$ называется *адекватным*, если для каждого $x \in \Gamma$ выполняется включение $S(x) \subseteq P'(x)$. Простейший пример адекватного языка — язык, в котором P представляет собой единичное разбиение Γ.

Для дальнейших рассуждений нам пригодятся следующие теоремы.

Теорема 1. Существует неадекватный язык.

Теорема 2. Если $\{\Gamma, P, \Phi\}$ — адекватный язык, то $R' = P'$.

Теорема 3. Пусть $\{\Gamma, P, \Phi\}$ — адекватный язык. Если классы $K(x)$ определяют разбиение K словаря Γ, то $K' = P'$.

Теорема 4. Если язык $\{\Gamma, P, \Phi\}$ однороден, то $K(x) = R(x)$ для каждого $x \in \Gamma$, т.е. классы совпадают со смешанными клетками.

4.3. Части речи и синтаксические типы

Рассмотрим язык $\{\Gamma, P, \Phi\}$, интерпретируя Γ как словарь естественного языка L, $P(x)$ (при $x \in \Gamma$) — как множество всех форм слова *x*, а Φ — как множество всех правильно построенных предложений языка L. Если слово x имеет две различные формы x_1 и x_2, то будем считать, что $P(x_1) \cap P(x_2) = 0$. Две омонимичные формы, такие, как *free* (прилагательное) и *free* (глагол), будут рассматриваться как различные слова, а соответствующие им P-клетки — как непересекающиеся.

При предлагаемой интерпретации клетки производного разбиения P' могут рассматриваться как приближенная модель частей речи в языке L.

Легко заметить, что для каждого $a \in \Gamma$ P-клетка $P(a)$ содержит только формы *a*. Для некоторых слов *a* $P(a)$ не содержит всех форм *a*, а только те, которые встречаются в предположениях из Φ, но в любом естественном языке выполняется одно очень простое условие: если два слова *a* и *b* принадлежат одной парадигме, т.е. если $b \in P(a)$, то они принадлежат одной части речи. В соответствии с этим правилом любое строгое определение понятия «часть речи» должно рассматривать любую часть речи как объединение P-клеток.

Пусть даны два слова *a* и *b*; мы можем считать, что $P(a)$ и $P(b)$ относятся к одной части речи тогда и только тогда, когда $P(a)$ и $P(b)$ P-эквивалентны, т.е. тогда и только тогда, когда $b \in P'(a)$.

Части речи не имеют абсолютного характера, они зависят от множества Φ отмеченных последовательностей (т.е. от синтаксиса языка), от разбиения P (т.е. от морфологии языка). Части речи в естественном языка являются функцией P и Φ, но мы никогда не учитываем при этом всех возможных предложений и парадигм данного языка. В каждом конкретном случае мы рассматриваем его фрагмент, достаточно сложный, чтобы получить хо-

рошее приближение к естественному языку, но в то же время достаточно простой, чтобы было возможно его систематическое и детальное исследование.

Объясним причины, побудившие дать частям речи приведенное выше определение. Рассмотрим язык $L = \{\Gamma, P, \Phi\}$. Этому языку мы можем поставить в соответствие другой язык, а именно $P(L) = \{\Gamma_1, P_1, \Phi_1\}$, где Γ_1 — множество всех P-клеток языка L, P_1 — разбиение Γ_1 на P-эквивалентные классы, а Φ_1 — множество всех отмеченных P-структур в языке L. Язык $P(L)$ будет называться *P-абстракцией языка* L. Данный уровень абстракции является именно таким, на котором становится понятной логическая структура частей речи, поскольку в $P(L)$ они не что иное, как дистрибутивные классы. Таким образом, P'-клетки языка L в точности соответствуют дистрибутивным классам языка $P(L)$.

Пусть дана часть речи **P**. Интересно найти класс **C** контекстов, для которых выполняются следующие условия: 1) для каждого слова $a \in \mathbf{P}$ существует контекст $(x, y) \in \mathbf{C}$ такой, что $xay \in \Phi$; 2) если дано слово b, которое не принадлежит **P**, то для такого слова не существует контекста $(u, v) \in \mathbf{C}$ такого, что $ubv \in \Phi$. Класс **C** называется *диагностическим классом* для **P**.

Выделение диагностических классов является очень важной задачей структурной лингвистики, поскольку диагностические классы для **P** позволяют нам выделить и изучать различные **P** по отдельности.

Выше не делалось никаких предположений о природе рассматриваемого языка. Но существует общая гипотеза, которая утверждает, что каждый естественный язык является адекватным.

Вспомним теорему 2 из предыдущего раздела. Если $\{\Gamma, P, \Phi\}$ — адекватный язык, то $R' = P'$. Эта теорема дает возможность по-новому определить понятие частей речи. Для данного слова $a \in \Gamma$ его часть речи совпадает с $R'(a)$. Еще одна возможность определения понятия «часть речи» в случае адекватного языка дается теоремой 3. Если в таком языке классы $K(x)$ определяют разбиение Γ, то $K' = P'$. Отсюда часть речи, которой принадлежит слово a, совпадает с $K'(a)$. Таким образом, становится ясно, что существуют адекватные неоднородные языки, в которых части речи могут быть построены тремя способами: как множество P'-клеток, R'-клеток и K'-клеток. Этот факт очень важен, поскольку в естественных языках суще-

ствует много фрагментов, которые являются адекватными, но не однородными.

Наиболее благоприятные условия в отношении анализа понятия «часть речи» предоставляют однородные языки. Действительно, согласно теореме 4 в любом однородном языке классы совпадают со смешанными клетками, т.е. $K(x) = R(x)$ для любого $x \in \Gamma$. Поскольку в любом адекватном языке для каждого $x \in \Gamma$ выполняется $R'(x) = P'(x)$, то, следовательно, в однородном языке $K'(x) = R'(x) = P'(x)$ для любого $x \in \Gamma$.

В соответствии с теоремой 4 можно построить части речи, взяв производное разбиение K'. Если *a* и *b* — два существительных различных родов, а *c* — прилагательное, то *K*-структуры $K(a)$ и $K(b)$ *K*-эквивалентны. Таким образом, $K(a)$ содержит все существительные. Если *a* — существительное, а *c* — прилагательное, то $K(a)$ и $K(c)$ не являются *K*-эквивалентными. Значит, части речи в том виде, как они описываются приведенной схемой, совпадают с традиционными.

Различия могут возникнуть, если иначе выбрать Γ и Φ; в особенности это касается местоимений, артиклей, числительных, наречий и некоторых типов прилагательных. Так, например, если не считать правильно построенным французским предложением такую последовательность слов, как *il est très mort*, то слово *mort*, не может быть отнесено к той же самой P'-клетке, что и *beau*.

Пусть даны два языка $L_1 = \{\Gamma_1, P_1, \Phi_1\}$ и $L_2 = \{\Gamma_2, P_2, \Phi_2\}$. Будем говорить, что они являются P'-*изоморфными*, если существует взаимно однозначное отображение φ словаря Γ_1 на Γ_2 такое, что $y \in \mathrm{P}_1'(x)$ в L_1 тогда и только тогда, когда $\varphi\ (y) \in \mathrm{P}_2'\varphi\ (x))$ в L_2. Поскольку части речи в точности совпадают с P'-клетками данного языка, то следовательно, P'-изоморфизм сохраняет подразделение на части речи.

Сущность проблемы синтаксических типов была сформулирована Ламбеком с помощью следующей аналогии: в классической физике можно просто проверить «грамматическую правильность» уравнения, сравнивая размерности его правой и левой частей. Спрашивается, нельзя ли подобным же образом приписывать грамматические типы словам в естественном языке так, чтобы грамматическая правильность предложения могла бы быть определена с помощью операций над этими типами.

Рассмотрим словарь *V*. Начнем с приписывания нескольких простейших типов отдельным словам и последовательностям над *V*. Из этих основных

типов с помощью трех формальных операций можно построить сложные типы. Такими операциями будут: *умножение*, *левое деление* и *правое деление*; первая из этих операций не имеет специального значка, а левое и правое деления обозначаются соответственно \ и /. Выражение $\mathrm{X} \to \mathrm{x}$ обозначает, что последовательность *X* относится к типу *x*. Сложные типы определяются следующим образом: если $\mathrm{X} \to \mathrm{x}$ и $\mathrm{Y} \to \mathrm{y}$, то $\mathrm{XY} \to \mathrm{xy}$; если $XY \to z$ и $Y \to y$, то $X \to z / y$ (читается *z над y*); если $XY \to z$ и $X \to x$, то $Y \to x \setminus z$ (читается *x под z*). Другими словами, выражение типа x / y, за которым следует выражение типа *y*, приводит к выражению типа *x*, так же как и выражение типа $y \setminus x$, которому предшествует выражение типа *y*.

Если любое выражение типа *x* принадлежит также типу *y*, то пишут $x \to y$. Из определения левого и правого делений следует, что и $(x / y) y \to x$, и $y(y \setminus x) \to x$.

В некоторых случаях, когда мы имеем дело с фрагментами естественных языков, мы можем рассматривать только два элементарных типа: *s* — тип предложений (т.е. отмеченных последовательностей) и *n* — именной тип. Но в более сложных случаях приходится учитывать значительно большее число элементарных типов. Так, иногда мы вводим в качестве элементарного типа *i* — инфинитив непереходного глагола. Для простоты ограничим здесь использование типа *s* и будем приписывать его только полным изъявительным предложениям (таким образом, мы исключаем большинство элементов диалога).

В наших примерах под именем будем понимать, прежде всего, собственные имена, а также такие выражения, которые могут встретиться в контекстах, аналогичных тем, в которых выступают собственные имена. Так, тип *n* приписывается так называемым неисчисляемым именам: *milk* «молоко», *rice* «рис» и т.д., которые могут встретиться без артикля, а также именным словосочетаниям типа *poor John* «бедный Джон» или *fresh milk* «свежее молоко». Но мы не можем приписать такой тип так называемым исчисляемым именам таким, как *king* «король», *chair* «стул» и т.д., которые требуют перед собой артикль, а также местоимению *he*, поскольку последнее не может заменять слова *John*, например, в предложении *poor John works*.

Рассмотрим словарь Γ и множество элементов, которые будем называть *элементарными типами*. Предположим теперь, что взята некоторая последовательность над Γ, и ее элементам приписаны определенные элементарные типы. Определим теперь множество Γ элементов, называемых *типами*, следующим образом: все элементарные типы являются типами; если *x* и *y* —

типы, то xy, $x \setminus y$ и x / y — также типы. Основные правила исчисления имеют следующий вид:

(1) $x \to x$;

(2) $(xy)z \to x(yz)$;

(2′) $x(yz) \to (xy)z$;

(3) если $xy \to z$, то $x \to z / y$;

(3′) если $xy \to z$, то $y \to x \setminus z$;

(4) если $x \to z / y$, то $xy \to z$;

(4′) если $y \to x \setminus z$, то $xy \to z$;

(5) если $x \to y$ и $y \to z$, то $x \to z$.

В соответствии с видом правил (2) и (2′) предлагаемое синтаксическое исчисление можно назвать *ассоциативным синтаксическим исчислением*. Его можно рассматривать абстрактно как формальный язык или как систему вывода. Естественно, есть дополнительное множество правил, которое имеет место в данном ассоциативном синтаксическом исчислении. Но некоторые правила здесь оказываются неверными. Например, следующие правила неверны: $(x / y) / z \to x /(y / z)$, $(x / y) \setminus z \to x /(y \setminus z)$, $xy \to yx$, $z \to (z / y)y$.

До сих пор типы сопоставлялись любым последовательностям слов, а не только тем, которые имеют завершенную форму. Предположим теперь, что мы приписываем типы не просто последовательностям слов, а грамматически правильным словосочетаниям, т.е. как бы расставляя в последовательности слов (и даже в последовательности морфем) скобки.

Рассмотрим множество последовательностей, называемых *атомарными словосочетаниями*, и примем следующее рекурсивное определение *словосочетания* (последовательное написание букв соответствует операции конкатенации, скобки имеют собственное значение, внешние скобки фразы могут быть опущены): все атомарные словосочетания являются словосочетаниями; если A и B — словосочетания, то (AB) — также словосочетание.

Типы вводятся с помощью аналогичного рекурсивного определения. Рассмотрим конечное множество, элементы которого называются *элементарными типами*. Все элементарные типы являются типами. Определены три бинарные операции над типами так, что если x и y — типы, то (xy), (x / y) и $(x \setminus y)$ — также типы.

Типы сопоставляются словосочетаниям по следующим правилам: если A относится к типу a и B — к типу b, то (AB) — к типу (ab); если (AB) относится к типу c и все B — к типу b, то A относится к типу (c/b); если (AB) относится к типу c, а все A — к типу a, то B относится к типу $(a\backslash c)$. Система основных правил для неассоциативного синтаксического исчисления имеет вид, аналогичный системе правил для ассоциативного синтаксического исчисления, за исключением того, что не выполняются правила $(xy)z \to x(yz)$ (2) и $x(yz) \to (xy)z$ (2′).

Мы можем теперь рассматривать неассоциативное синтаксическое исчисление как систему вывода. Рассмотрим сначала множество элементов, которые называются *переменными*. Затем рекурсивно определим еще одно множество, элементы которого назовем *термами*: все переменные являются термами; если x и y — термы, то (xy), $(x\backslash y)$ и (x/y) — также термы. Введем единственную *формулу*: $x \to y$ (где x и y — термы), единственную *аксиому*: $x \to x$ и правила вывода (3), (3′), (4), (4′) и (5). Синтаксическое исчисление позволяет нам перевести некоторые грамматические правила из грамматики в словарь.

Предположим, что мы заменили все грамматические правила языка, сопоставив подходящим образом типы единицам словаря. При этом становится возможным проанализировать данную последовательность слов автоматически. Расставляя скобки, мы выделяем в последовательности словосочетания и подписываем под каждым словом один из типов, сопоставленных ему в словаре. Пусть x — сложный тип, соответствующий всей последовательности. Если имеем $x \to s$, то X — утвердительное предложение, если $x \to i$, то X — повелительное предложение, и т.д. Этот процесс повторяется для всех вариантов расстановки скобок и всех вариантов приписывания словам типов.

Приведем формальное определение категориальных грамматик. *Двунаправленной категориальной грамматикой* называется кортеж $\langle \Gamma, C, \Sigma, R, f \rangle$ из следующих пяти элементов: Γ — конечное множество элементов (словарь), C — замыкание конечного множества основных категорий, обозначим их $\psi_1, ..., \psi_n$ относительно операции левого и правого делений (таким образом, если α и β — категории, то α/β и $\alpha\backslash\beta$ — также категории), Σ — выделенная в множестве C категория (категория предложений), R — множество, состоящее из двух правил сокращения: $(\varphi_i/\varphi_j)\varphi_j \to \varphi_i$ и

$\varphi_i(\varphi_i \setminus \varphi_j) \to \varphi_j$, f — функция, сопоставляющая элементам из Γ конечные подмножества из C.

Мы будем говорить, что последовательность категорий β *непосредственно сокращается* до α, если β является результатом применения к α одного из правил сокращения. Мы будем также говорить, что α *сокращается* до β, если β является результатом применения конечного числа правил сокращения к α; более точно, если существует последовательность категорий $\gamma_1,...,\gamma_n$ такая, что $\alpha = \gamma_1$, $\beta = \gamma_n$ и γ_i непосредственно сокращается до γ_{i+1} при $i = 1, \ ..., \ n-1$.

Последовательность x над Γ называется *предложением* тогда и только тогда, когда по крайней мере одна последовательность категорий, приписанных элементам x в соответствии с f, сокращается до Σ. Множество всех таких предложений образует *язык, задаваемый данной двунаправленной категориальной грамматикой*. Такой язык называется *двунаправленным категориальным языком*.

4.4. Грамматический род

Грамматический род представляет собой одну из наиболее интересных проблем теории грамматики. Исследовались различные аспекты этой проблемы, такие как связь между значением существительного и его родом; соотношение между родом существительного и его окончанием. Род рассматривался в свете соответствия между планом содержания и планом выражения с точки зрения синтаксических и контекстуальных связей, с точки зрения его происхождения и эволюции.

Почти все авторы сходятся на том, что семантических критериев явно недостаточно для понимания сложной природы грамматического рода и необходимо использовать все лингвистические факты, относящиеся к этой категории.

Попытаемся сначала объяснить (не строгим образом, а в расчете на интуитивное понимание), какова формальная природа соотношения между естественным и грамматическим родом. Возьмем в качестве исходного два существительных ξ и η. Хотим найти такое определение мужского и женского грамматических родов, которое бы в явной форме представило тот механизм, те операции, посредством которых эти категории получены из соответствующих естественных родов. Будем исходить из того, что любое существительное мужского рода должно находиться в строго определенном

формальном отношении с существительным ξ, и любое существительное женского рода должно находиться в подобном же отношении с существительным η. Чтобы получить такое отношение, обратимся к понятию цепи, введенному Ревзиным.

Если специально не оговаривается противное, то во всех последующих примерах множество предложений будет состоять лишь из синтагм типа существительное + качественное прилагательное в положительной степени или качественное прилагательное в положительной степени + существительное.

Понятия цепи и длины цепи позволяют получить такое определение мужского и женского грамматических родов, которое предполагает переход от естественного к грамматическому роду. Правила, представленные далее, объясняют многие конкретные факты естественных языков (на примере французского языка).

Существительное относится к мужскому грамматическому роду, если любая словоформа его парадигмы может быть соединена с любой словоформой парадигмы слова ξ *посредством цепи, длина которой не превышает 3*. Из этого непосредственно следует, что и ξ принадлежит мужскому грамматическому роду.

Существительное относится к женскому грамматическому роду, если любая словоформа его парадигмы может быть соединена с любой словоформой парадигмы слова η *посредством цепочки, длина которой не превышает 3*. Отсюда непосредственно следует, что η принадлежит женскому грамматическому роду.

Существительное относится к среднему роду, если оно не относится ни к женскому, ни к мужскому грамматическому роду.

Существительное относится к общему роду, если оно принадлежит как мужскому, так и женскому роду.

Эти правила характеризуют категорию грамматического рода во многих естественных языках. Считается, что эта категория является невырожденной для существительных данного языка в том случае, когда в этом языке существует, по крайней мере, одно существительное мужского рода, которое нельзя отнести к женскому роду, и, по крайней мере, одно существительное женского рода, которое нельзя отнести к мужскому роду. Рассмотрим приведенные выше правила на примере французского языка.

Пусть $\xi = \text{homme}$, $\eta = \text{femme}$. Существительные *colins*, *cahier*, *murs* и т.д. относятся к мужскому роду, так как имеется цепь: (а) *colins*, *colin*,

homme, (b) *colins*, *hommes*, (c) *colin*, *homme*, *hommes*, (d) *cahier*, *homme*, (e) *cahiers*, *hommes*, *homme*, (f) *cahier*, *homme*, *hommes*, (g) *murs*, *mur*, *homme*, (h) *murs*, *hommes*, (i) *mur*, *homme*, *hommes*; существительные *plumes*, *pluie*, *feuile* и другие принадлежат женскому роду, так как любая словоформа из парадигм этих существительных может быть соединена с любой формой слова *femme* цепью, длина которой не больше 3; имеются цепи *plumes*, *plume*, *femme*; *pluie*, *femme*, *femmes*; *feuille*, *femme* и т.д. Существительные *cas*, *tas*, *tapis*, *nez*, *voix* и другие, у которых форма единственного числа совпадает с формой множественного числа, являются существительными среднего рода, так как нет ни одной формы слова *homme* и ни одной формы слова *femme*, которые принадлежали бы тому же дистрибутивному классу, что и *cas* (имеются формы *petit cas*, *petits cas*, но нет ни *petit hommes*, ни *petits homme*), так же как не существует никакой формы *homme* и никакой формы *femme*, которые принадлежали бы тому же дистрибутивному классу, что и *voix* (имеются формы *belle voix*, *belles voix*, но нет ни *belle femmes*, ни *belles femme*; имеются *homme beau*, *hommes beaux*, но нет ни *voix beau*, ни *voix beaux*). Существительные *camarade*, *élève*, *enfant* и другие, которым могут предшествовать как прилагательные в форме мужского рода, так и женского, относятся к среднему роду, так как дистрибутивные классы таких слов не совпадают с дистрибутивными классами ни словоформ *homme*, ни словоформ *femme*. Действительно, имеются *bon élève*, *bonne élève*, но нет ни *bon femme*, ни *bonne homme*; имеются *bons élèves*, *bonnes élèves*, но отсутствуют *bons femmes* и *bonnes hommes*.

Итак, во французском языке имеются существительные, которые относятся к мужскому роду и не относятся к женскому (*cahier*, *mur*, *soleil* и т.д.), а также существительные, которые относятся к женскому роду, но не относятся к мужскому (*plume*, *pluie*, *feuille* и т.д.); значит, грамматический род французских существительных не является вырожденным.

Сделаем следующие замечания.

1. Некоторые незначительные противоречия связаны с ограниченным запасом контекстов (прилагательное + существительное, существительное + прилагательное). Если соответствующим образом преобразовать исходное множество отмеченных последовательностей, то подобные противоречия исчезнут.

2. Приписывание словам грамматических родов не абсолютно однозначно. Оно зависит от определенного выбора парадигм и предложений (отмеченных последовательностей слов). Если мы, например, возьмем только изолированные слова в качестве предложений, то все они образуют единственный дистрибутивный класс, так что, взяв любые два слова, мы

можем соединить их цепью, длина которой равна 2. В частности, все существительные будут относиться как к мужскому, так и к женскому роду, и следовательно, грамматический род будет вырожденным. Это замечание раскрывает синтагматический характер грамматического рода, его существенную зависимость от контекста.

Пусть $\{\Gamma, P, \Phi\}$ — некоторый язык. Будем говорить, что два слова $a \in \Gamma$ и $b \in \Gamma$ *принадлежат одному и тому же роду*, и будем обозначать это $a \gamma\ b$, если для всякого $a' \in P(a)$ и всякого $b' \in P(b)$ выполняется, по крайней мере, одно из двух следующих условий: $P(a) \cap S(b') \neq 0$, $P(b) \cap S(a') \neq 0$. Будем говорить, что a и b *принадлежат одному и тому же ограниченному роду*, и будем записывать это $a \rho\ b$, если для всякого $a' \in P(a)$ и всякого $b' \in P(b)$ верно, что $P(a) \cap S(b') \neq 0 \neq P(b) \cap S(a')$.

Становится ясно, что две словоформы могут иметь один и тот же род, даже если они не принадлежат одной и той же части речи. Из-за этого факта возникает противоречие между грамматическим родом и его математической моделью, потому что имеет смысл говорить о роде только для отдельных частей речи. Например, грамматический род у прилагательных совсем иной, чем у существительных; наши модели имеют отношение лишь к существительным.

Перейдем теперь к проблеме измерения различий между двумя данными родами. С этой целью введем понятие *расстояния между двумя родами* $G(x)$ и $G(y)$, которое определяется, как наименьшее число n такое, что всякое слово рода $G(x)$ может быть соединено со всяким словом рода $G(y)$ цепью, длина которой не более $n + 1$. В том случае, когда такого числа не существует, мы будем говорить, что расстояние между $G(x)$ и $G(y)$ бесконечно. Будем обозначать через $\delta(x, y)$ расстояние между $G(x)$ и $G(y)$. Нетрудно видеть, что $\delta(x, y) \geq 0$, $\delta(x, y) = \delta(y, x)$, $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$ для всякого слова z. Это понятие особенно удобно, когда множества слов, относящихся к разным родам, не пересекаются. Рассмотрим понятие расстояния между родами на примере французского языка.

Рассмотрим семь следующих родов: G(*crayon*) — первый мужской, G(*arbre*) — второй мужской, G(*maison*) — женский, G(*cas*) — первый средний, G(*voix*) — второй средний, G(*enfant*) — третий средний и G(*camarade*) — четвертый средний. Поскольку *bel arbre* и *bels arbres* — правильные слово-

сочетания в отличие от *bel crayon*, *bels crayons*, то $\delta(crayon, arbre) = \infty$. С другой стороны, так как *beau crayon* и *beaux crayons* — правильные словосочетания, а *beau maison* и *beaux maisons* таковыми не являются, то $\delta(crayon, maison) = \infty$. Словосочетания *nouveau cas* и *nouveaux cas* правильные, а *nouveaux crayon* и *nouveau crayon* неправильные, поэтому $\delta(crayon, cas) = \infty$. Поскольку *beau voix* и *beaux voix* неправильны, то $\delta(crayon, voix) = \infty$. Ввиду того что правильные словосочетания *belle enfant* и *belles enfants* становятся неправильными, если заменить *enfant* на *crayon* и *enfants* на *crayons*, $\delta(crayon, enfant) = \infty$. Поскольку *bel arbre*, *bels arbres* правильно, тогда как *bel cas* и *bels cas* неправильно, то $\delta(arbre, cas) = \infty$. Ввиду того что словосочетания *bel voix*, *bels voix* неправильны, то $\delta(arbre, voix) = \infty$. Ввиду того что *belle enfant* и *belles enfants* правильны, тогда как *belle arbre*, *belles arbres* правильными не являются, $\delta(arbre, enfant) = \infty$. Поскольку *grande voix* и *grandes voix* — правильные словосочетания, а *grande maisons* и *grandes maison* — нет, то $\delta(maison, voix) = \infty$. Поскольку *bel enfant* и *bels enfants* являются правильными словосочетаниями, тогда как *bel maison* и *bels maisons* — нет, то $\delta(maison, enfant) = \infty$. Подобным же образом мы находим, что $\delta(maison, cas) = \infty$, $\delta(cas, voix) = \infty$ и $\delta(voix, enfant) = \infty$. Из того, что словосочетания *beau camarade*, *beaux camarades*, *bell camarade* и *belles camarades* являются правильными, тогда как *beaux camarade*, *beau camarades*, *belles camarade*, *belle camarades*, *bel camarade*, *bels camarades* неправильны, следует, что имеют место равенства

$$\delta(crayon, camarade) = \delta(arbre, camarade) = \delta(maison, camarade) =$$
$$\delta(cas, camarade) = \delta(voix, camarade) = \delta(enfant, camarade) = \infty.$$

Нетрудно видеть, что два различных рода никогда не пересекаются.

4.4. Категории падежа

Пусть Γ — фиксированное множество элементов, называемых словами, P — разбиение множества Γ и Φ — подмножество свободной полугруппы над Γ. Элементами этой полугруппы являются последовательности слов, а элементами Φ — отмеченные или маркированные последовательности. Предполагается, что каждое слово входит хотя бы в одну отмеченную по-

следовательность. Если $x \in \Gamma$, то $P(x)$ — это элемент разбиения P, содержащий x; $y \in P(x)$ будет пониматься как форма слова x. $P(x)$ будет *парадигмой* слова x. Как и прежде, тройка $\{\Gamma, P, \Phi\}$ будет называться *языком с парадигматической структурой*, или просто *парадигматическим языком*.

Контекстом мы будем называть упорядоченную пару последовательностей (f_1, f_2), причем последовательности f_1 и f_2 не обязательно являются отмеченными. Контекст (f_1, f_2) называется *отмеченным*, если существует такое слово x, что последовательность $f_1 x f_2$ является отмеченной.

Слово a является *допустимым* в контексте (f_1, f_2), если существует такое слово $a' \in P(a)$, что последовательность $f_1 a' f_2$ отмечена.

Слово *a непосредственно допустимо* в контексте (f_1, f_2), если последовательность $f_1 a f_2$ является отмеченной. Очевидно, что всякое слово, непосредственно допустимое в контексте (f_1, f_2), является одновременно допустимым в (f_1, f_2), однако обратное неверно. Обозначим через $A(\gamma)$ множество слов, допустимых в контексте $\gamma = (f_1, f_2)$.

Два контекста γ' и γ'' будем называть *эквивалентными*, если $A(\gamma') = A(\gamma'')$. Пусть $B(\gamma)$ — множество слов, непосредственно допустимых в контексте γ. Очевидно, что $B(\gamma) \subseteq A(\gamma)$. Обратное не всегда верно.

Два контекста γ' и γ'' *строго эквивалентны*, если $B(\gamma') = B(\gamma'')$. Нетрудно понять, что два строго эквивалентных контекста эквивалентны.

Пусть **E** — подмножество Γ. Будем говорить, что **E** — *допустимое множество*, если существует хотя бы один такой контекст γ, что $\mathbf{E} = A(\gamma)$.

Назовем **E** *непосредственно допустимым*, если существует хотя бы один такой контекст γ, что $\mathbf{E} = B(\gamma)$. Будем говорить, что множества **E** и **H** являются *одновременно допустимыми*, если существует такой контекст γ, что $\mathbf{E} = A(\gamma)$ и $\mathbf{H} = B(\gamma)$.

Мы будем говорить, что два контекста γ' и γ'' *согласованы относительно* $a \in \Gamma$, если существует такое слово $a' \in P(a)$, что a' непосредственно допустимо одновременно и в γ', и в γ''.

Мы будем говорить, что γ' и γ'' *полностью согласованы относительно* $a \in \Gamma$, если $P(a) \cap B(\gamma') = P(a) \cap B(\gamma'') \neq 0$.

Очевидно, что отношения согласования и полного согласования относительно слова а симметричны на множестве контекстов языка.

Введем еще два новых понятия. Пусть даны два контекста γ_1 и γ_2. Мы будем говорить, что γ_1 и γ_2 *конгруэнтны*, и будем обозначать это $\gamma_1 \approx \gamma_2$, если существует конечная цепочка контекстов $\delta_1, \delta_2, ..., \delta_i, \delta_{i+1}, ..., \delta_n$, обладающая тремя следующими свойствами: (1) $\delta_1 = \gamma_1$; (2) $\delta_n = \gamma_2$; (3) для любого $i < n$ контексты δ_i и δ_{i+1} согласованы.

Если в третьем условии вышеприведенного определения заменить термин «согласованы» на «полностью согласованы», то мы получим новое отношение. В этом случае мы будем говорить, что γ_1 и γ_2 *полностью конгруэнтны*, и будем обозначать это $\gamma_1 \tilde{\approx} \gamma_2$.

Очевидно, что два согласованных контекста являются конгруэнтными и что два полностью согласованных контекста полностью конгруэнтны. Обратное утверждение неверно.

Отмеченные контексты образуют разбиение на классы конгруэнтности, т.е. на классы эквивалентности относительно $\approx$. Каждый класс конгруэнтности на множестве отмеченных контекстов мы будем называть *грамматическим падежом* или просто *падежом*.

Отсюда ясно, что каждый отмеченный контекст соотнесен с каким-либо грамматическим падежом, причем только с одним. Отмеченные контексты также образуют разбиение относительно полной конгруэнтности. Всякий класс полной конгруэнтности мы назовем совмещенным грамматическим падежом.

Тем не менее, можно отметить следующее несовершенство модели. Контексты (*я вижу синий*, θ) и (*синий*, θ) являются согласованными, а значит, и конгруэнтными, хотя с точки зрения традиционной грамматики мы имеем здесь разные падежи. Чтобы обойти эту трудность, можно отказаться от некоторых контекстов с прилагательными, порядковыми числительными и т.п.

К числу недостатков модели относится и то, что разные словоформы одного и того же существительного могут быть непосредственно допустимыми в одном контексте. Например, формы *газеты* и *газету* непосредственно допустимы в контексте (не читал, θ), формы *кошке* и *кошку* непосредственно допустимы в контексте (дал, θ). Чтобы избежать этого, аналогично ограничивают классы рассматриваемых контекстов таким образом, чтобы две различные словоформы одного и того же слова не могли быть непосредственно допустимыми в одном контексте.

5. СЛОВООБРАЗОВАТЕЛЬНЫЕ КОНСТРУКЦИИ

В данном разделе приведены основные словообразовательные конструкции русского языка, *суффиксация* от основ прилагательных и глаголов и другие методы. В частности, этими способами образуются большинство существительных.

5.1. Словообразование имен существительных

Для *префиксальных* существительных наиболее характерны следующие словообразовательные категории.

1. Производные с общим значением интенсивности, высокой степени того, что названо производящей основой:

раз-: *красавица — раскрасавица, граф — разграф;*
сверх-: *человек — сверхчеловек, прибыль — сверхприбыль;*
супер-: *экспресс — суперэкспресс, кубок — суперкубок;*
архи-: *демократ — архидемократ, порядок — архипорядок;*
ультра-: *мода — ультрамода, звук — ультразвук;*
гипер-: *вежливость — гипервежливость, фонды — гиперфонды;*
экстра-: *мода — экстрамода, совершенство — экстрасовершенство.*

2. Производные с общим значением противоположности, отрицания:

анти-: *частица — античастица, герой — антигерой;*
противо-: *меры — противомеры, действие — противодействие;*
контр-: *удар — контрудар, позиция — контрпозиция, инициатива — контриниициатива;*
де-/дез-: *гуманизация — дегуманизация; информация — дезинформация;*
не-: *счастье — несчастье, покой — непокой;*
дис-: *пропорция — диспропорция, гармония — дисгармония, комфорт — дискомфорт.*

3. Производные с общим значением неистинности, ложности:

псевдо-: *рецензия — псевдорецензия, открытие — псевдооткрытие;*
квази-: *диалог — квазидиалог, открытие — квазиоткрытие.*

4. Производные со значением совместности:

со-*: автор — соавтор, переживание — сопереживание.*

5. Производные со значением подчиненности:

под-*: вид — подвид, группа — подгруппа, язык — подъязык;*

суб-*: продукты — субпродукты, инспектор — субинспектор, аренда — субаренда.*

Кроме того, в словообразовании существительных действуют отдельные малопродуктивные приставки.

Префиксально-суффиксальным способом образуются существительные на базе сочетаний существительных с предлогами. В составе производного предлог преобразуется в приставку. Производящим является сочетание имени существительного с предлогом. Производные, как правило, обозначают предметы, соотнесенные в пространственном или временном отношении с тем, что названо производящей основой. Наиболее употребительны при этом суффиксы *-ник, -j(e), -иц(а)*.

1. Производные с пространственными значениями:

под-…-ник (предмет, находящийся под тем, что названо производящей основой): *под снегом — подснежник, под локтем — подлокотник;*

на-…-ник (предмет, находящийся на том, что названо производящей основой): *на колене — наколенник, над почкой — надпочечник, на конце — наконечник;*

за-…-j(e) (то, что находится за предметом, названным производящей основой): *за рекой — заречье, за Волгой — Заволжье;*

при-…-j(e) (то, что находится рядом с предметом, названным производящей основой): *при Урале — Приуралье, при море — приморье;*

меж-…-j(e) (то, что находится между предметами, названными производящей основой): *меж горами — межгорье, меж бровями — межбровье*.

2. Производные с временными значениями:

меж-…-j(e): *меж сезонами — межсезонье;*

пред-…-j(e): *пред зимой — предзимье.*

3. Производные, обозначающие отсутствие того, что названо производящей основой, образуются с помощью приставки *без-* и суффиксов *-иц(а)* и *-j(e)*:

без-…-иц(а)*: без работы — безработица;*

без-…-j(e)*: без денег — безденежье, без людей — безлюдье, без земли — безземелье, без культуры — бескультурье*.

Группа префиксально-суффиксальных существительных, которая не соотносится с сочетанием предлог + существительное. Приставка, входящая в

производное, не имеет омонимичного предлога. Эта группа малочисленна и дает мало новообразований. В производных участвуют приставка *анти-* и суффикс *-ин*: *комар — антикомарин*, шутливые: *антиведмин, антилюбвин*. Производные называют средство против того, что названо основой производящего существительного.

Префиксация в соединении с нулевой суффиксацией используются для производства от основ имен прилагательных существительных со значением «слабой степени обнаружения какого-либо признака, названного производящей основой». При этом действует префикс *про-* в соединении с нулевым суффиксом.

Производные содержат те же чередования, что слова типа *сушь*, и относятся к тому же склонению: *желтый — прожелть, седой — проседь*.

Образование сложных существительных происходит с помощью сложения, а также слиянием способа сложения с префиксальным, суффиксальным и префиксально-суффиксальным способами.

5.2. Словообразование имен прилагательных

В словообразовании имен прилагательных развиты суффиксация, префиксация, префиксально-суффиксальный способ, словосложение и сращение.

Префиксация — один из самых распространенных способов словообразования прилагательных.

Наиболее многочисленна категория прилагательных, обозначающих интенсивность, полноту проявления признака:

наи- (прилагательное обычно содержит суффикс *-ейш-/-айш-*): *умнейший — наиумнейший, глупейший — наиглупейший;*

пре-: *умный — преумный, глупый — преглупый;* чаще в повторах: *добрый-предобрый, милый-премилый;*

раз-: *веселый — развеселый;* чаще в повторах: *удалый-разудалый;*

пере- (обычно в повторах): *тертый-перетертый, писанный-переписанный;*

архи-: *прозаический — архипрозаический, благородный — архиблагородный;*

сверх-: *молодой — сверхмолодой, одаренный — сверходаренный;*

супер-: *модный — супермодный, тяжелый — супертяжелый;*

экстра-: *новый — экстрановый, модный — экстрамодный;*

ультра-: *оригинальный — ультраоригинальный, современный — ультрасовременный*.

Другую словообразовательную категорию составляют прилагательные, в которых приставки имеют значение отрицания, противоположности:

не-*: умный — неумный, красивый — некрасивый;*

анти-*: гуманный — антигуманный, военный — антивоенный;*

противо-*: воспалительный — противовоспалительный* («направленный против воспаления»), *химический — противохимический, атомный — противоатомный* («направленный против химического или атомного оружия»).

Префиксально-суффиксальным способом образуются прилагательные на базе именных и глагольных сочетаний.

Прилагательные, производимые на базе сочетаний существительных с предлогами, преобразуемыми в составе производных в приставки, могут включать разнообразные приставки, а из суффиксов *-н-* (преимущественно), реже *-ов-, -ск-* и некоторые другие:

без-*: без пилота — беспилотный, без шума — бесшумный;*

вне-*: вне завода — внезаводской, вне земли — внеземельный;*

внутри-*: внутри института — внутриинститутский, внутри школы — внутришкольный;*

до-*: до школы — дошкольный, до посева — допосевный;*

за-*: за бортом — забортовой, за рекой — заречный;*

меж-*: меж планетами — межпланетный, меж районами — межрайонный;*

на-*: на горе — нагорный, на столе — настольный;*

над-*: над глазом — надглазный, над окном — надоконный;*

около-*: около звезды — околозвездный, около солнца — околосолнечный;*

под-*: под вагоном — подвагонный;*

после-*: после полета — послеполетный, после посева — послепосевный;*

пред-*: перед съездом — предсъездовский, перед дипломом — преддипломный;*

противо-*: против пожара — противопожарный;*

чрез-*: через полосу — чересполосный.*

На базе соединения отрицания *не* с сочетанием имени существительного с предлогом *без* образуются имена прилагательные со сложной приставкой *небез-* и суффиксом *-н-*. Производное имеет значение неполноты, слабой степени проявления признака: *не без пользы — небесполезный, не без греха — небезгрешный, не без вреда — небезвредный*. Прилагательные с приставкой *небез-* отличаются по значению от прилагательных с приставкой *без-* и

отрицанием *не-*. Например: *Этот напиток не безвреден* (то есть вреден, отрицается безвредность напитка). — *Этот напиток небезвреден* (то есть вреден в слабой степени).

Прилагательные, образованные префиксально-суффиксальным способом от основ глаголов, образуются с помощью приставки *не-* и суффиксов *-н-* и *-м-*. Производные обозначают «невозможность подвергнуться действию»:

не-…-н(ый)*: разорвать — неразрывный, возвратить — невозвратный;*

не-…-м(ый)*: исцелить — неисцелимый, переносить — непереносимый.*

Префиксация в соединении с нулевой суффиксацией используется для производства прилагательных на базе сочетаний предлога *без* с именами существительными. Обычно эти существительные называют часть тела человека или животного. Производное обозначает «не имеющий того, что названо производящей основой»: *без ноги — безногий, без руки — безрукий*. Реже производящая основа относится к иному семантическому разряду: *без листьев — безлистый, без коры — безкорый*. Возможны однокорневые образования с тем же значением, содержащие с своем составе суффикс: *безлистый — безлистный, безъязыкий — безъязычный*.

5.3. Словообразование глаголов

В словообразовании глаголов главенствующее место занимает внутриглагольное словообразование, т. е. производство от основ глагола чистой префиксацией и префиксально-суффиксальным способом. Суффиксация имеет меньшее значение, а словосложение для словообразования глагола совсем мало характерно.

Среди префиксальных глаголов различаются две большие группы: с приставками пространственных значений и с приставками количественно-временных значений.

Глаголы с приставками пространственных значений обозначают различные направления действия:

в-*: бежать — вбежать, нести — внести;*

вз-*: бежать — взбежать* (в гору), *лететь — взлететь;*

до-*: бежать — добежать, лететь — долететь;*

за-*: бежать — забежать* (за что-нибудь), *лететь — залететь* (за что-нибудь, куда-нибудь);

на-*: бежать — набежать, лететь — налететь;*

о-/об-*: бежать — обежать, лететь — облететь* (что-нибудь);

от-*: бежать — отбежать, лететь — отлететь;*

пере-*: бежать — перебежать, лететь — перелететь* (через реку);
при-*: бежать — прибежать, лететь — прилететь;*
про-*: бежать — пробежать* (под мостом), *лезть — пролезть;*
с-*: бежать — сбежать* (с горы), *лезть — слезть* (со стола);
у-*: бежать — убежать , лететь — улететь.*

Глаголы с приставками количественно-временных значений обозначают временные пределы, а также силу, интенсивность или слабость, неполноту действия.

1. Глаголы, обозначающие начало процесса:
за-*: петь — запеть, говорить — заговорить;*
по-*: бежать — побежать, плыть — поплыть;*
вз- (с оттенком интенсивности действия)*: кричать — вскричать, реветь — взреветь.*

2. Глаголы, обозначающие окончание процесса:
от- («прекращение процесса»): цвести **—** *отцвести, греметь — отгреметь;*
до- («доведение действия до конца»): *думать — додумать, читать — дочитать, шить — дошить.*

3. Глаголы, обозначающие окончание действия с оттенками полноты, тщательности, энергичности, силы его выполнения:
раз-*: ругать — разругать, баловать — разбаловать, целовать — расцеловать;*
пере-*: жарить — пережарить, кормить — перекормить;*
на-*: безобразничать — набезобразничать, возить — навозить:*
про-*: жарить — прожарить, варить — проварить* (как следует)*, сушить — просушить;*
вы-*: мерить — вымерить, строгать — выстрогать.*

4. Глаголы, обозначающие полную исчерпанность предмета действием, а также причинение неприятности, ущерба действием:
вы-*: топтать — вытоптать, рубить — вырубить;*
из-: *мылить — измылить* (все мыло), *писать — исписать* (карандаш);
за-*: сечи — засечь* (до смерти), *играть — заиграть* (мелодию);
с-*: носить — сносить* (башмаки), *винтить — свинтить* (винт);
от-*: сидеть — отсидеть* (руку), *топтать — оттоптать* (ноги);
про-*: носить — проносить* (брюки до дырки), *плакать — проплакать* (все глаза).

5. Глаголы, обозначающие дополнительное, добавочное действие, добавление чего-либо действием, а также слабость, неполноту действия:

до- («дополнительное действие»): *грузить — догрузить, купить — докупить;*

под- («дополнительное действие — для количественного увеличения объекта», «прибавление действием»): *лить — подлить, купить — подкупить;*

под- («дополнительное, повторное действие — для более полного достижения результата»): *варить — подварить* (кашу; = доварить), *красить — подкрасить;*

при- (неполнота действия): *встать — привстать, лечь — прилечь;*

над-*: рубить — надрубить, колоть — надколоть, бить — надбить;*

недо-*: выполнить — недовыполнить, думать — недодумать, рассказать — недорассказать.*

Префиксально-суффиксальным способом образуются глаголы от основ глаголов и от основ других частей речи.

Глаголы, производимые от основ глаголов, образуют две группы:

а) приставка + производящая основа + суффикс несовершенного вида;

б) приставка + производящий глагол + — *ся*.

Рассмотрим каждую группу более детально.

1. Неполноту, ослабленность действия обозначает группа одновидовых глаголов несовершенного вида, имеющих прерывисто-длительное значение:

по-*: трещать — потрескивать, кричать — покрикивать;*

при- (с оттенком дополнительности действия): *плясать — приплясывать, говорить — приговаривать;*

под- (с оттенком дополнительности действия): *петь — подпевать, свистеть — подсвистывать;*

на-*: петь — напевать, свистеть — насвистывать, шептать — нашептывать.*

Интенсивность, тщательность совершения действия выражает группа глаголов с приставками:

вы-*: шагать — вышагивать, плясать — выплясывать;*

на-*: звонить — названивать, хвалить — нахваливать.*

2. Глаголы, образуемые одновременным присоединение приставки и *-ся*, многообразны по значениям:

в-…-ся («предельная исчерпанность действия, углубленность в действие»): *читать — вчитаться, думать — вдуматься;*

за-…-ся («увлеченность действием»): *думать — задуматься, читать — зачитаться;*

на-…-ся («удовлетворенность действием»): *гулять — нагуляться, работать — наработаться;*

из-…-ся («исчерпанность в результате действия, приобретение каких-либо отрицательных свойств в результате действия»): *хулиганить — исхулиганиться, писать — исписаться;*

до-…-ся («достижение результата после интенсивного выполнения действия»): *звать — дозваться, звонить — дозвониться.*

Глаголы, производимые от основ имен существительных префиксально-суффиксальным способом, используют различные приставки в соединении с суффиксами *-∅ -/-и(ть)* и *-е(ть)*:

за-…ить: *штора — зашторить;*

об-…-ить: *лес — облесить;*

при-…-ить: *земля — приземлить*.

Для отыменного производства глаголов префиксально-суффиксальным способом характерен префикс *обез-*. Производные на *-ить* обозначают «лишить того, что названо производящей основой»: *вред — обезвредить*. Производные на *-еть* обозначают «лишиться того, что названо производящей основой»: *сила — обессилеть, ум — обезуметь*.

Глаголы, производимые от основ имен прилагательных суффиксально-префиксальным способом, производятся с помощью различных приставок (чаще всего *у-* и *о-*) и тех же суффиксов *-∅ -/-и(ть)* и *-е(ть)*: *плотный — уплотнить, глубокий — углубить, благородный — облагородить, слабый — ослабеть, скудный — оскуднеть*.

5.4. Наречие как неизменяемая часть речи и типы наречий

Наречия — это такие неизменяемые полнозначные слова, которые способны называть признак действия или признак признака. Есть наречия, способные называть признак предмета: *яйцо всмятку, разговор по-английски* и наречия, называющие признак предмета или признак признака действия: *очень красивое лицо, шли очень долго.* Наречия не могут определяться прилагательными.

5.4.1. Семантические разряды наречий

Из определения наречий следует, что их синтаксическая сочетаемость весьма разнообразна. С различием в синтаксической сочетаемости связано и различие в семантических разрядах. Наречия со значением меры и степени *(много, очень, совсем, чуть-чуть)* в принципе могут относиться к глагольным формам, к имени прилагательному и наречию. Однако при наличии семантической общности каждое наречие из этой группы обладает собст-

венными сочетательными свойствами (ср.: *много говорить — очень разговорчивый; едва видеть — едва видный*).

Наречия также могут иметь значение качественной характеристики действия: *громко кричать*, *чисто писать*; образа действия: *ехать верхом*, *смотреть исподлобья*; времени: *прийти вовремя*, *сказать завтра*; места: *смотреть кругом*, *оглянуться назад*; причины: *взять сослепу*, *сказать сгоряча*; цели: *сделать назло*, *написать нарочно*, *сказать умышленно*. Наречия данной семантики сочетаются обычно с глагольными формами, однако не полностью исключена и их сочетаемость с существительными (ср.: *взглянуть вперёд — взгляд вперёд*).

Практически полезным является правило: если при проверяемом слове может быть определяющее его прилагательное — перед нами существительное; если появление прилагательного невозможно — перед нами наречие. Например, *шёл берегом,* но *шёл высоким, красивым левым берегом,* следовательно, *берегом* — падежная форма существительного; *волосы ёжиком* — вставка прилагательного практически невозможна, следовательно, *ёжиком* — наречие.

Среди наречий, как и среди имен, есть местоименные слова: *там, потом, когда, так, зачем* и неместоименные: *медленно, громко, назло* и др.

5.4.2. Степени сравнения наречий

Многие наречия имеют синтетические образования со значением большой степени качества: *смело — смелее (смелей), внимательно — внимательнее (внимательней* или *внимательнейше).* Степени сравнения наречий создают разные лексемы.

Сравнительная стенень прилагательных и наречий совпадает по форме и разграничить эти образования можно лишь по синтаксическим признакам. Если сравнительная степень характеризует действие, это — сравнительная степень наречия. Если сравнительная степень характеризует предмет, это — сравнительная степень прилагательного. Например: *Петя пишет* ***красивее,*** *чем Ваня* — наречие; *Петя* ***красивее*** *Вани* — прилагательное.

5.4.2. Категория состояния и слова различных групп в синтаксической функции сказуемого

Состав категории состояния довольно широк. Это происходит, во-первых, из-за рассмотрения слов типа *скучно, весело, тоскливо,* употребляемых в функции сказуемого как категории состояния. В итоге получается, что в слове типа *скучно* мы имеем три омонима, принадлежащих к разным

частям речи: *Это занятие* ***скучно*** — краткое прилагательное; *Он рассказывал долго и* ***скучно*** — наречие; *Мне* ***скучно*** — категория состояния.

Во-вторых, категория состояния включает в себя такие употребляемые в функции сказуемого слова, как *грех, срам, пора, время, лень.* Таким образом, слово *грех* в предложении ***Грех*** *— не беда, молва нехороша*, является существительным, а в следующем предложении : *Над старостью смеяться* ***грех*** — категорией состояния.

Можно сказать, что категория состояния — это такие слова, которые, не будучи глаголами, выступают в функции сказуемого. При этом уже не обращается внимание на то, являются ли эти слова изменяемыми, выполняют ли они только функцию сказуемого.

5.4.3. Объем группы слов, принадлежащих к категории состояния

К категории состояния, как правило, относят лишь неизменяемые слова. Все изменяемые слова — существительные, прилагательные, — даже употребленные в функции сказуемого, остаются соответственно существительными и прилагательными.

Сложнее обстоит дело со словами *пора, лень, грех, срам.* Когда эти слова выполняют функцию сказуемого, они меняют свою родовую характеристику, согласуясь с формами глагола прошедшего времени среднего рода. Ср.: *Ужасная пора (лень); Ужасный грех (срам)* и *Пора (лень, грех, срам) было так поступать.* Это обстоятельство — аргумент в пользу признания слов *пора, лень, грех, срам,* употребляемых в функции сказуемого, словами категории состояния.

Среди полнозначных неизменяемых слов можно выделить:

1) слова, способные выполнять функцию определителя признака или действия: *очень, быстро, неожиданно;*
2) слова, способные выполнять указанную функцию и функцию сказуемого: *светло, грустно, весело;*
3) слова, способные выполнять функцию сказуемого: *нужно, можно, жаль.*

К категории состояния следует относить только слова третьей группы. Рассматривая вторую группу, важно всякий раз оценивать, в какой именно синтаксической функции выступает слово. Однако из различия в этой функции не следует делать прямолинейный вывод о различиях в принадлежности к частям речи. Следует иметь в виду, что многие слова второй группы выступают преимущественно в функции сказуемого.

Итак, категорию состояния как часть речи составляют неизменяемые полнозначные слова, единственная синтаксическая функция которых — функция сказуемого.

5.4.4. Модальные слова

Среди самостоятельных неизменяемых слов есть такие, которые вообще не бывают каким-либо членом предложения, но способны образовать слова-предложения: *вероятно, конечно, видимо* и т. п. Эти слова нельзя рассматривать ни как существительные, ни как наречия или категорию состояния.

Кроме синтаксической общности, самостоятельности и неизменяемости, данные слова обладают и общностью номинативной. Номинативная общность состоит в том, что с помощью этих слов выражается отношение говорящего к действительности, а также к форме и содержанию высказывания об этой действительности. Таким образом, в составе модальных слов оказываются не только слова типа *вероятно,* но и такие слова, как *во-первых, итак, следовательно*.

Различают несколько разрядов модальных слов по их значению и употреблению:

1) слова, указывающие на чужие мысли, слова, стиль или их оценку: *буквально, так сказать;*
2) слова, указывающие на эмоциональную оценку действительности: *спасибо, полно, пожалуй;*
3) слова, указывающие на достоверность высказывания: *несомненно, безусловно, очевидно, видимо, в самом деле, подлинно;*
4) слова, указывающие на последовательность или характер связи мыслей или событий: *кроме того, в частности, в конце концов, во-первых, кстати.*

Исходя из приведенных определений наречия и категории состояния, представляется логичным рассматривать в качестве особой части речи неизменяемые знаменательные слова — модальные слова, не способные быть каким-либо членом предложения. При этом особенно важно отличать модальные слова как часть речи от других частей речи, употребленных в качестве вводных слов. Например, в предложениях *К сожалению примешивалось чувство облегчения* и *К сожалению, поездка не состоится* мы имеем предложно-падежную форму существительного как дополнение и как вводное слово, но перехода из одной части речи в другую не происходит. То же самое наблюдается и в предложениях *Он пришёл кстати* и *Кстати, я получил вчера письмо*.

5.5. Служебные части речи

Служебные части речи противопоставляются самостоятельным знаменательным (по-другому, — полнозначным) частям речи на основании принципа «функция в предложении».

Те слова, которые являются членами предложения и/или могут быть употреблены самостоятельно в качестве слова-предложения, формируют самостоятельные части речи. Слова, лишенные обеих таких возможностей, принадлежат к служебным частям речи.

Так, полнозначные слова *зима, молчу, хорошо* могут быть членами предложения и могут образовать предложения. Модальные слова типа: *безусловно, во-вторых, пожалуй,* могут образовать предложения, однако не могут быть членами предложения. Относимые обычно к модальным словам языковые единицы типа *мол, дескать,* не могут быть членами предложения и не могут образовать предложения.

Служебные части речи делят на две группы по принципу «наличие или отсутствие функции выразителя связи между членами предложения или предложениями». Одни служебные слова связывают части предложения: *под, к, с, а, так как.* Другие служебные слова непосредственно связующих функций не выполняют: *бы, вот.*

Выполнение служебными словами связующей функции никак не означает отсутствия у них какого-либо номинативного значения. Указывая на наличие связи, служебные части речи одновременно определяют, каковы именно содержательные отношения между связываемыми единицами. Например, в словосочетаниях *шёл* ***в*** *метро, шёл* ***от*** *метро, шёл* ***к*** *метро* служебные слова не только указывают на то, что словоформы *шёл* и *метро* связаны между собой, но и несут в каждом случае определенную конкретную информацию о направлении действия относительно предмета. Ср. также: *видеть и знать — видеть, чтобы знать; не работал, но устал — не работал, потому что устал.*

Служебные части речи несколько напоминают окончания, поскольку многие окончания также одновременно и связывают слова в предложении, и отражают определенный аспект действительности.

Все служебные части речи являются неизменяемыми, у них отсутствуют какие-либо морфологические характеристики. Поэтому классификация внутри служебных частей речи может быть проведена только на основании принципов семантических и синтаксических.

5.5.1. Союз как служебная часть речи

Союз — это такая часть речи, к которой относятся слова, используемые для связи словоформ в простом предложении и частей сложного предложения, либо только для связи частей сложного предложения.

Например, союз *и* может связывать и словоформы в предложении: *Отец и сын,* и части сложного предложения: *Прозрачный лес один чернеет, и ель сквозь иней зеленеет*, а союз *чтобы (чтоб)* может соединять только части предложения: *Он рыбачил тридцать лет и три года и не слыхивал,* ***чтоб*** *рыба говорила*.

Союзы принято делить на непроизводные: *а*, *но*, *и*, *как* и производные: *чтобы*, *оттого*, *потому что*, *в то время как* и др. Производные союзы могут быть простые: *чтобы*, *оттого* и составные: *после того, как*; *вследствие того, что*; *для того, чтобы*; *несмотря на то, что*; *как только*.

Составные союзы, на первый взгляд, выступают как словосочетания. Во всяком случае, они пишутся в виде нескольких слов и представляют собой такую последовательность, части которой похожи на самостоятельные лексемы. Однако данные языковые единицы не обладают релевантными признаками словосочетаний. «Части» рассматриваемых единиц имеют строго фиксированный порядок следования — вставить что-либо или невозможно: *потому что, в то время как,* или, наоборот, совершенно необходимо: *не только—но и, если—то, то...то*. Выделяются составные союзы, части которых совершенно различны: *не только—но и; если—то,* и союзы, части которых совпадают: *то-то, ни-ни* (их не следует путать с повторяющимися простыми союзами). Ср.: *А за окном* ***то*** *дождь,* ***то*** *снег*; *И пращ, и стрела, и лукавый кинжал щадят победителя годы* — в первом примере наличие первого *то* требует и второго *то;* во втором примере появление любого *и* теоретически совершенно независимо от наличия или отсутствия других *и*.

5.5.2. Классификация союзов

Как следует из определения союзов, среди них выделяются слова, которые соединяют однородные члены предложения, части сложносочиненных предложений и самостоятельные предложения. Такие союзы называют сочинительными: *и, а, но, или* и др.

Есть и такие слова, которые соединяют главную и придаточную части сложноподчиненного предложения. Такие союзы называют подчинительными: *после того как, ввиду того что, хотя* и др.

Как исключение в простом предложении подчинительные союзы могут связывать словоформы, которые не являются однородными членами пред-

ложения. Таков, например, союз *как,* присоединяющий сравнительный оборот или употребляющийся в значении «в качестве»: ***Как*** *невесту, родину мы любим*; *В городе он известен* ***как*** *хороший врач.* Другим исключением такого же типа является союз *чем* в составе оборота, включающего сравнительную степень: *Брат старше,* ***чем*** *сестра.*

Семантическая классификация союзов частично «перекрещивается» с их синтаксической классификацией. Сочинительные союзы по значению делят на соединительные, противительные, разделительные, градационные, пояснительные.

Соединительные союзы выражают отношения перечисления: *Пришёл,* ***и*** *ослабел,* ***и*** *лёг под сводом шалаша на лыки*; противительные — отношения различия, несоответствия: *Поклажа бы для них казалась и легка,* ***да*** *лебедь рвётся в облака, рак пятится назад,* ***а*** *щука тянет в воду*; разделительные — отношения взаимоисключения: ***Или*** *пан,* ***или*** *пропал*; градационные, называя явления, выделяют одно из них как более важное: *Он* ***не только*** *читает,* ***но и*** *хорошо говорит по-французски* (ср.: *Он по-французски совершенно мог изъясняться и писал*); пояснительные (*а именно*, *как-то*, *или*) указывают на различие в именовании того же самого явления.

Подчинительные союзы по значению делят на семантические, т. е. передающие определенные смысловые отношения, и асемантические, т. е. чисто синтаксические, указывающие лишь на «подчиненность» придаточной части сложноподчиненного предложения (*как, что, чтобы*). Семантические подчинительные союзы могут иметь значение времени: *когда, пока, едва, лишь, только, после того как* и др.; условия: *если, раз, ежели* и др.; причины: *потому что, так как, ибо* и др.; уступки: *хотя; несмотря на то, что* и др.; цели: *чтобы, затем чтобы* и др.; следствия: *так что* и сравнения: *как, словно, точно.*

5.5.3. Предлог как служебная часть речи

Предлог — это такая служебная часть речи, которая выражает различные «подчиненные» отношения существительного к другим словам в словосочетании и предложении. Предлоги, как и другие служебные части речи, принадлежат к неизменяемым словам.

По степени словообразовательной сложности выделяют предлоги непроизводные: *в, до, за, на, от, по, при, с* и производные: *вокруг, благодаря, в течение, навстречу* и др.

По признаку морфологической сложности среди производных предлогов следует выделить простые предлоги: *путём, спустя* и составные: в *связи с, независимо от, судя по* и др. Несмотря на особенности правописания,

составные предлоги представляют собой последовательности с жестким порядком следования частей.

Предлоги могут выражать различного рода отношения:

– пространственные (*к, от, под, над, за, в, из, на* и др.);

– временные (*после, в течение* и др.);

– причинные (*из-за, по причине* и др.).

Реже встречаются некоторые другие отношения, выражаемые предлогами. Как и союзы, предлоги могут выступать в чисто синтаксической функции. Например, в словосочетаниях *смеяться над бедой, верить в успех, надеяться на лучшее, бороться с недостатками* предлоги лишены какой-либо семантической функции, их присутствие обусловлено исключительно синтаксическими свойствами глаголов.

Многие предлоги многозначны. Эта многозначность, как и у других частей речи, устраняется благодаря контексту. Например: *шел **из** дома* — пространственное значение, *делал **из** зависти* — причинное.

Предлоги требуют, чтобы «подчиненное» им существительное было употреблено в определенной падежной форме. Поэтому предлоги делятся на такие, после которых непременно следует один падеж, либо два или даже три.

Р.п.	Д.п.	В.п.	Т.п.	П.п.
без	к	про	над	при
от	благодаря	сквозь	перед	
около	согласно	через		
из-за	вопреки			
	по			

Предлог *между* требует существительного в родительном или творительном падежах, предлог *в* — в винительном и предложном, предлог *под* — в винительном и творительном.

5.5.4. Частицы

Частицы — это такие служебные слова, которые выражают смысловые и модально-экспрессивные оттенки предложений и слов и участвуют в образовании форм слова. Как и другие служебные слова, частицы не имеют форм словоизменения.

В настоящее время по признаку «функция» принято выделять частицы, участвующие в выражении грамматических значений наклонения совмест-

но с другими элементами глагольной формы (речь идет о сослагательном и повелительном наклонениях глагола и соответственно о частицах *бы, да, давай, пусть, пускай),* и частицы, имеющие лексические значения.

Частицы, имеющие лексические значения, подразделяются на несколько семантических разрядов: отрицательные частицы *не* и *ни.* Среди субъективно-модальных значений частиц выделяют усилительные: ***Даже*** *он пришёл; Он* ***же*** *знал об этом;* выделительные: ***Только*** *он пришел;* ***Лишь*** *он знал об этом;* вопросительные: ***Неужели*** *он пришёл?;* ***Разве*** *он знал об этом?;* восклицательные: ***Ведь*** *он пришёл!; Он* ***куда как*** *хорошо знал об этом?;* указательные: *вот, вон, это;* определительно-уточняющие: *именно, точно, приблизительно, почти;* утвердительные: *действительно, конечно, точно;* выражающие передачу чужой мысли: *мол, дескать* и некоторые другие. В этой классификации нет четкой границы между модальными (полнозначными) словами и частицами, поскольку некоторые слова, упомянутые выше как частицы (служебные слова), могут образовать слова-предложения и, следовательно, могут рассматриваться как модальные слова.

5.5.5. Междометия и другие слова вне частей речи

Среди неизменяемых лексем особое место занимают междометия, слова-предложения, передающие, хотя и не называющие конкретно, различные чувства, такие, например, как удивление, испуг, гнев, радость: ***Ба,*** *знакомые всё лица*; ***Ага,*** *попались!*

Сам характер эмоций подчас не зависит прямо от лексического значения междометия, а возникает в результате контекстуальных воздействий. Например, междометие *ах* вне контекста может означать и радость, и испуг, и разочарование. Неизменяемость является чрезвычайно существенным свойством междометий. Изменяемые лексемы, обладающие теми же семантическими свойствами, уже не междометия. Так, в предложении *Он никак не мог привыкнуть ко всем этим* ***охам и ахам*** слова «*ох*» и *«ах»* — существительные. Междометиями являются и сложные (состоящие не из одного корня) образования. Например, *боже мой, чёрт возьми* — междометия, каждое из которых представляет собой одну лексему, хотя графически в каждой из них выделяются две словоформы.

К междометиям относят и безаффиксные глагольные образования типа: *прыг, толк: Татьяна* ***прыг*** *в другие сени*; *Мартышка... тихохонько медведя* ***толк*** *ногой*. Строго говоря, у слов типов *ах, ба* и *прыг* мало общего; с междометиями их объединяет морфологическая общность, а именно неизменяемость.

К междометиям иногда относят и звукоподражательные слова типа *мяу-мяу, кис-кис.* Эти слова, не обозначая эмоций, по своим семантическим свойствам явно отличаются от слов типа *ах.* По семантическим и синтаксическим свойствам они противостоят и словам типа *прыг.* Считают, что звукоподражательные слова являются словами вне частей речи.

5.6. Местоименные слова

5.6.1. Категория рода и одушевленности—неодушевленности у местоименных существительных

К местоименным существительным относят местоимения личные: *я, ты, мы, вы, он, она, оно;* возвратное: *себя;* вопросительные: *кто, что;* неопределенные: *кто-то, что-то, кто-нибудь, что-нибудь, кто-либо, что-либо, кое-кто, кое-что, некто, нечто;* отрицательные: *никто, ничто, некого, нечего.*

Хотя местоименные существительные редко сочетаются с прилагательными, только по их сочетанию, как известно, можно определить родовую принадлежность существительного.

В соответствии с этим критерием местоименные существительные *он, кто-то, кто-нибудь, кто-либо, кое-кто, некто* принадлежат к мужскому роду; *она* — к женскому; *оно, что, что-то, что-нибудь, что-либо, кое-что, нечто* и *ничто — к* среднему.

Местоименные существительные *я, ты* принадлежат к общему роду: *я говорил (говорила), ты сказал (сказала).* Местоимение *кто* может сочетаться с формами прилагательных мужского и женского рода, указывая соответственно на лиц мужского и (или) женского пола. Однако при обозначении лиц обоего пола местоимение *кто* сочетается с глаголом в формах прошедшего времени и сослагательного наклонения лишь в форме мужского рода: *Кто вышел замуж?*

Местоименное существительное *себя* может, видимо, иметь не одну родовую форму; ср.: *себя, больного; себя, больную.* Местоименные существительные *некого, нечего* можно также причислять и к мужскому и к среднему роду.

Все местоименные существительные обладают категорией одушевленности—неодушевленности: я, *ты, мы, вы, кто* и производные от него — одушевленные; *что* и производные от него — неодушевленные. Местоименные существительные *он, она, оно, они, себя* могут выступать и как одушевленные, и как неодушевленные: *(вижу) их всех — их все.* Одушев-

ленность или неодушевленность местоименных существительных зависит в этом случае от одушевленности или неодушевленности заменяемых местоимениями существительных: *вижу их всех — альпинистов,* но *вижу их все — вершины.*

5.6.2. Категория числа у местоименных существительных

Противопоставление по числу внутри парадигмы имеют местоименные существительные *он, она, оно.* Формы множественного числа у этих лексем омонимичны.

Личные местоименные существительные *я, ты* и *мы, вы* нельзя рассматривать как связанные только противопоставлением по числу. Ведь *мы* — это не «много я», но «я и другие»; *вы* — это не «много ты», а «совокупность различных лиц».

Остальные местоименные существительные по своим семантическим свойствам не чужды идее счета, но не обладают морфологическими средствами для выражения числового противопоставления. Они напоминают существительные типа *пальто* и *сани.* При этом местоименные существительные *себя, кто, кто-то, кто-нибудь, кто-либо, кое-кто* в принципе более или менее естественно могут выражать числовые противопоставления согласуемыми словоформами: *себя, больного — себя, больных; кто такой — кто такие* (ср.: *новое пальто — новые пальто).*

Другие не чуждые идее счета местоименные существительные не способны выражать противопоставление по числу и самой своей формой и согласуемыми словоформами: *что такое* возможно, а *что такие* — невозможно (ср.: *новые сани* — и об одних санях и о нескольких санях).

5.6.3. Категория падежа у местоименных существительных

У некоторых местоименных существительных нет части падежных форм. Не существует формы именительного падежа у местоименных существительных *себя, некого, нечего.*

Местоименное существительное *некто,* наоборот, имеет лишь эту форму. Только форму именительного-винительного падежа имеет *нечто.* В тех случаях, когда "предметы", обозначенные этими местоимениями, оказываются в контекстах, требующих иных падежных форм, приходится использовать иные лексемы: *нечто произошло* (им. п.), *увидел нечто* (вин. п.), но *приблизился к чему-то* (дат. п.), *подумал о чём-то* (предл. п.).

Кроме того, последовательное «открытие» падежей для местоименных существительных *он, она, оно* должно привести к увеличению числа паде-

жей. Местоименные существительные *он, она, оно* в ряде падежей, выделенных для существительных, могут иметь по две формы, отнюдь не находящихся между собой в отношениях свободного варьирования.

Речь идет о случаях типа *у него нет* (род. п.) и *лишиться его* (род. п.), *подошёл к ней* (дат. п.) и *нравится ей* (дат. п.), *вижу их* (вин. п.) и *смотрю на них* (вин. п.) и т. д. Несколько упрощая дело, можно утверждать, что после предлогов выступают формы с начальным *н,* при отсутствии предлогов — формы, не имеющие *н*.

Наличие разных форм характеризует не только местоименные существительные *он, она, оно, они,* но и *никто, ничто, некого, нечего,* у которых в сочетании с предлогом представлены так называемые разорванные формы: *не лишился ничего (никого); нет ни у чего (ни у кого); сказать некому — пойти не к кому.*

5.6.4. Местоименные прилагательные

Значительную группу лексем среди традиционно выделяемых местоимений составляют местоименные прилагательные. По своей семантике они подразделяются на притяжательные: *мой, твой, наш, ваш, свой;* указательные: *тот, этот, такой, этакий, таков, следующий;* определительные: *всякий, всяческий, каждый, любой, весь, целый, иной, другой, сам, самый;* вопросительные: *какой, который, чей, каков;* неопределенные: *какой-то, какой-нибудь, какой-либо, чей-то, чей-нибудь, чей-либо, кое-какой, некоторый, некий;* отрицательные: *никакой, никоторый, ничей.*

Все местоименные прилагательные — изменяемые, они обладают словоизменительными категориями рода, числа и падежа. Местоименные прилагательные лишены противопоставления по полноте—краткости. Все они полные в том смысле, что изменяются не только по числам и родам, но и по падежам. В отличие от обычных прилагательных не все местоименные прилагательные могут выступать в роли сказуемого.

5.6.5. Местоименные числительные

Местоименные слова выделяются и среди числительных: *сколько, столько, несколько, сколько-нибудь, сколько-то, нисколько.* Так же, как и иные числительные, эти слова не обладают категориями рода и числа.

Проявление категорий падежа у этих лексем такое же, как у обычных числительных: они «подчиняют» себе существительные в именительном падеже (*несколько друзей),* равном ему винительном (*вижу несколько до-*

мов) и в дательном падеже с предлогом *по* в распределительном значении: *По сколько рублей они получили?*

Во всех остальных падежах местоименные числительные согласуются в падеже со связанным с ними существительным: *к нескольким представителям, говорить о стольких вещах.* У местоименных числительных в винительном падеже может проявляться категория одушевленности — неодушевленности: *(вижу) несколько домов — нескольких юношей.* Впрочем, это противопоставление не всегда выдерживается: можно сказать и *вижу несколько юношей.* По семантике местоименные числительные могут быть подразделены на вопросительные: *сколько;* указательные: *столько;* отрицательные: *нисколько;* неопределенные: *несколько, сколько-то* и т. п.

5.6.6. Склонение местоименных слов

Достаточно полное описание типов склонения местоименных слов обычно представлено в грамматических словарях русского языка.

Нерегулярное склонение имеют местоименные существительные *я, ты, мы, вы, себя:*

И.	*я*	*ты*	*мы*	*вы*	
Р., В.	*меня*	*тебя*	*нас*	*вас*	*себя*
Д.	*мне*	*тебе*	*нам*	*вам*	*себе*
Т.	*мной*	*тобой*	*нами*	*вами*	*собой*
П.	*мне*	*тебе*	*нас*	*вас*	*себе*

Склонение местоименных существительных *кто, что* приближается к склонению прилагательных типа *лисье* или *дядино.* Местоименные существительные *он, она, оно* имеют различные основы для именительного падежа обоих чисел и для всех иных падежей. Окончания таких лексем практически совпадают с окончаниями прилагательных типа *лисий.* Особенность данных лексем состоит в том, что в зависимости от наличия или отсутствия предлогов они имеют разные падежные формы.

Склонение местоименных существительных *никто, ничто,* а также *некого, нечего* (обладающих дефектом в парадигме) управляется двумя правилами: первое относится к склонению местоименных существительных *кто, что,* второе касается различий в формах одного падежа в зависимости от наличия или отсутствия предлога.

И.		*он, оно*	*она*	*они* (мн. ч.)
Р.	без предл.	*его*	*её*	*их*
	с предл.	*от него*	*от неё*	*от них*
Д.	без предл.	*ему*	*ей*	*им*
	с предл.	*к нему*	*к ней*	*к ним*
В.	без предл.	*его*	*её*	*их*
	с предл.	*про него*	*про неё*	*про них*
Т.	без предл.	*им*	*ею (ей)*	*ими*
	с предл.	*с ним*	*с ней (нею)*	*с ними*
П.		*о нем*	*о ней*	*о них*

Среди местоименных прилагательных выделяется несколько разновидностей склонения, напоминающих склонение прилагательных типа *дядин* или *лисий.*

6. МИНИМАЛЬНЫЕ СХЕМЫ ПРЕДЛОЖЕНИЙ

Расширенные схемы предложения = минимальные схемы + не входящие в них конструктивные, то есть существенные для семантической структуры предложения, компоненты.

Для описания объективного содержания предложения в современных исследованиях используются разные понятия, например, понятие пропозиции, заимствованное из логики. Структуру пропозиции определяет предикат, который несет в себе признак предмета или отношение между предметами, указывает определенные места для предметов — актантов, определяет их количество и роли.

Предикаты могут быть нульместными, одноместными и многоместными. Их валентные свойства связаны с семантическими и определяются ими. Так, предикаты **состояния и характеристики** лица, животного или предмета — одноместны. Их единственный актант всегда имеет значение **субъекта состояния**: *Он тревожится (тревожен, в тревоге); Ему холодно*.

Предикаты **состояния окружающей среды**, нульместны: *Светает; Заря; Морозно; Идет дождь*.

Предикаты **активного действия** могут обозначать **процессы**, то есть действия, не направленные на изменение ситуации, в которой они протекают: *Дети бегают; Все работают* — в этом случае, они одноместны; а мо-

гут обозначать **события**, то есть действия, направленные на изменение ситуации, предполагающие «исходное» и «завершающее» ее состояния: *Художник нарисовал картину; Мы сажали капусту; Я взял у соседа книгу* — в этом случае они многоместны.

Количество мест также зависит от семантики предиката, равно как и от роли актантов. Так, предикаты со значением **давания—получения** трехместные: *Сосед дал мне книгу*.

Ниже перечисляются минимальные, т.е. простейшие, схемы предложений.

Двухкомпонентные номинативные

1. Существительное (в ИП) + Глагол (его спрягаемая форма).
   N_1V_f . Например: *Грачи прилетели; Зеленеют деревья; Все дела делаются людьми.*

С помощью предикативной конструкции этой схемы можно выразить пропозиции с одноместным или двухместным предикатом, имеющим значение активного целенаправленного действия (то есть здесь можно рассматривать одноместные или двухместные предикаты указанного выше типа): *Здесь он работает; Утром он разносит газеты.*

По этой схеме могут быть оформлены пропозиции с одноместным предикатом характеризующей семантики: *Он лгун; Он добряк.*

2. Существительное (в ИП) + Глагол-связка «быть» (его спрягаемая форма) + Прилагательное или Страдательное Причастие (их полная форма в ИП или ТП, или краткая форма или компаратив).
   $N_1Cop_fAdj_{f\ /1/5}$. Например: *Ночь была тиха (тихая, тихой); Через час был объявлен привал; Машины готовы к испытаниям; Он ранен.*

По этой схеме могут быть оформлены пропозиции с одноместным предикатом характеризующей семантики: *Он лжет; Он хромает.*

3. Существительное (в ИП) + Глагол-связка «быть» (его спрягаемая форма) + Существительное (в ИП или ТП).
   $N_1Cop_fN_{1/5}$. Например: *Он был студент (студентом); Орел — хищник; Это наше общежитие.*

По этой схеме могут быть оформлены пропозиции с одноместным предикатом характеризующей семантики: *Он лживый; Он хромой; Он добрый.*

4. Существительное (в ИП) + Глагол-связка «быть» (его спрягаемая форма) + Существительное (в беспредложной или предложной

форме любого косвенного падежа (все падежи, кроме ИП и ТП), которая способна сочетаться с глаголом-связкой «быть») или Наречие (способное сочетаться с глаголом-связкой «быть»).

$N_1 Cop_f N_{2...pr}/Adv_{pr}$. Например: *Этот дом будет без лифта; Мы были в отчаянии; Чай — с сахаром; Приход Ивана Ивановича был кстати; Все были начеку; У него глаза навыкате.*

Двухкомпонентные инфинитивные

5. Глагол (инфинитив) + Глагол (его спрягаемая форма).
   $InfV_f$. Например: *Не мешало б нам встречаться чаще (Светлов); Отмалчиваться не следует; Курить воспрещалось; Быть космонавтом (смелым) хочется каждому мальчишке; Друзьям разрешалось быть вместе.*
6. Глагол (инфинитив) + Глагол-связка «быть» (его спрягаемая форма) + Прилагательное или Страдательное Причастие (их полная форма в ИП или ТП, или краткая форма или компаратив).
   $InfCop_f Adj_{f\ /1/5}$. Например: *Промолчать было разумно (разумнее, самое разумное, самым разумным); Уговаривать его было излишне (излишнее, излишним); Нужно уехать; Правильнее было бы признать свою ошибку.*
7. Глагол (инфинитив) + Глагол-связка «быть» (его спрягаемая форма) + Существительное (в ИП или ТП).
   $InfCop_f N_{1/5}$. Например: *Дозвониться — проблема (было проблемой); Главной его целью было (главная его цель была) увидеть все своими глазами; Строить — это радость; Любить иных — тяжелый крест (Пастернак); Оказывается, быть взрослой — не всегда преимущество (Нагибин); Превосходная должность быть на земле человеком (Горький).*
8. Глагол (инфинитив) + Глагол-связка «быть» (его спрягаемая форма) + Существительное (в беспредложной или предложной форме любого косвенного падежа, которая способна сочетаться с глаголом-связкой «быть») или Наречие (способное сочетаться с глаголом-связкой «быть»).
   $InfCop_f N_{2...pr}/Adv_{pr}$. Например: *Промолчать было не в его правилах; Купить машину нам не по средствам; Молчать некстати; Идти дальше было невмоготу.*

9. Глагол (инфинитив) + Глагол-связка «быть» (его спрягаемая форма) + Глагол (инфинитив).

   $\mathrm{Inf Cop_f Inf}$. Например: *Отказаться было обидеть; Быть студентом — это постоянно учиться мыслить; Быть актером — прежде всего быть талантливым человеком.*

Однокомпонентные

10. Глагол (его форма единственного числа третьего лица или среднего рода).

    $\mathrm{V_{s3/\ n}}$. Например: *Скрипело, свистало и выло в лесу (Заболоцкий); Смеркается; Ему нездоровится; Дохнуло свежестью; Крышу охватило пламенем; Пароход покачивало; У него накипело на сердце; Об этом уже писалось.*

Схема предполагает изображение семантического признака нецеленаправленности, непроизвольности действия, несвязанности его с сознательной деятельностью человека.

11. Глагол (его форма множественного числа третьего лица).

    $\mathrm{V_{pl3}}$. Например: *За столом зашумели; Его обидели; Здесь о молодых специалистах заботятся, им доверяют; Во время еды не разговаривают.*

Эта схема изображает действие как сознательное, целенаправленное и потому исходящее обязательно от человека, но лицо, от которого исходит это действие, остается «за кадром». Поэтому она не допускает употребления глаголов, обозначающих действия и состояния, не связываемые с человеком (*годиться, зазеленеть, случаться, отразиться, состоять из*).

Здесь можно рассматривать одноместные или двухместные предикаты, имеющие значение активного целенаправленного действия: *Здесь работают; Утром разносят газеты.*

12. Глагол-связка «быть» (его форма единственного числа третьего лица или среднего рода) + Прилагательное или Страдательное Причастие (их краткая форма или компаратив единственного числа среднего рода).

    $\mathrm{Cop_{s3/\ n} Adj_{fsn}}$. Например: *Было темно; Морозно; Ночью будет холодно; Душно без счастья и воли (Некрасов).*

По этой схеме строятся предложения, передающие разнообразные состояния.

13. Глагол-связка «быть» (его форма единственного числа третьего лица или среднего рода) + Существительное (в беспредложной или предложной форме любого косвенного падежа, которая способна сочетаться с глаголом-связкой «быть») или Наречие (способное сочетаться с глаголом-связкой «быть»).
    $\mathrm{Cop}_{s3/\,n}\mathrm{N}_{2\,\ldots\,pr}/\mathrm{Adv}_{pr}$. Например: *Было уже заполночь; Завтра будет без осадков; Нам не до сна; Ей было невдомек; Пусть будет по-твоему; Ему не к спеху.*
14. Глагол-связка «быть» (его форма множественного числа третьего лица) + Прилагательное или Страдательное Причастие (их краткая форма или компаратив множественного числа).
    $\mathrm{Cop}_{pl3}\mathrm{Adj}_{fpl}$. Например: *Ему были рады; Им довольны; Отказом были обижены.*
15. Глагол-связка «быть» (его форма множественного числа) + Существительное (в беспредложной или предложной форме любого косвенного падежа, которая способна сочетаться с глаголом-связкой «быть») или Наречие (способное сочетаться с глаголом-связкой «быть»).
    $\mathrm{Cop}_{pl}\mathrm{N}_{2\,\ldots\,pr}/\mathrm{Adv}_{pr}$. Например: *Дома были в слезах; От него были в восторге; С ним были запросто.*
16. Глагол-связка «быть» (его спрягаемая форма) + Существительное (в ИП).
    $\mathrm{Cop}_{f}\mathrm{N}_{1}$. Например: *Шепот. Робкое дыханье. Трели соловья (Фет); Тишина; Была зима.*

Здесь можно рассматривать одноместные или двухместные предикаты, имеющие значение активного целенаправленного действия: *Здесь ему работать; Утром ему разносить газеты.*

17. Глагол (инфинитив).
    Inf. Например: *Не нагнать тебе бешеной тройки (Некрасов); Только детские книги читать. Только детские думы лелеять (Мандельштам); Быть рекам чистыми; Быть мальчишке поэтом; Быть по-вашему; Всем быть в спортивной форме.*

Здесь можно рассматривать одноместные или двухместные предикаты, имеющие значение активного целенаправленного действия: *Здесь у него работа; Утром у него разноска газет.*

СПИСОК ЛИТЕРАТУРЫ

1. **Апресян Ю.Д.** Экспериментальное исследование семантики русского глагола. — М.: Наука, 1967. — 251 с.
2. **Мельчук И.А.** Опыт теории лингвистических моделей типа «Смысл⇔Текст». — М. 1974. — 315 с.
3. **Маркус С.** Теоретико-множественные модели языков. — М.: Наука, 1970. — 332 с.
4. **Современный** русский язык: Учеб. для филол. спец. высших учебных заведений/ Под ред. В.А. Белошапковой. — М.: Азбуковник, 1997. — 928 с.
5. **Современный** русский язык: Учеб. для филол. спец. высших учебных заведений/ Под ред. Д.Э. Розенталя. — М.: Изд. МГУ, 1971. — 636 с.

Т.В. Батура, Ф.А. Мурзин

ЛОГИЧЕСКИЕ МЕТОДЫ ПРЕДСТАВЛЕНИЯ СМЫСЛА ТЕКСТА НА ЕСТЕСТВЕННОМ ЯЗЫКЕ*

ВВЕДЕНИЕ

В рамках реализуемого проекта предполагается разработать методы, которые позволят проводить разносторонний анализ текстов и отдельных предложений на естественном языке. Предполагается использовать такие методы, как представление смысла текста в рамках подхода И.А. Мельчука и предложенные им лексические функции [1], методы из работ Апресяна [2], теоретико-множественные модели Маркуса [3]. Предполагается также адаптировать для целей изучения текстов на естественном языке некоторые методы и конструкции математической логики: конструкцию Генцена, применяемую в теореме о существовании модели и в теоремах об опускании типов [4], конечный форсинг и т.д.

Целью данной работы является разработка разнообразных алгоритмов сопоставления предикатов и формул узкого исчисления предикатов текстам на естественном языке. Также осуществляется попытка сопоставления конечных моделей предложениям текста и тексту целиком.

Полученные результаты в дальнейшем могут быть подвергнуты изучению и различным преобразованиям средствами математической логики, что даст возможность осуществить переход с синтаксического на семантический уровень и в некоторой степени «научить» машину понимать смысл текста на естественном языке.

Результаты работы могут быть применены в автоматизированных системах акцепции информации из текстов на естественном языке, интеллектуальных системах поиска информации в сети, при построении систем автоматического резюмирования, электронных переводчиков и словарей.

Данная работа может оказаться полезной при создании поисковых систем: в случаях, когда по запросу из документа нужно извлечь необходимую информацию или по заданному запросу из большого количества документов извлекать релевантные, т.е. соответствующие данному запросу, документы. На основе этой работы возможно создание системы, способной реконструировать содержание и выделять имеющиеся в тексте знания, кото-

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

рые могут быть представлены пользователю в виде компактных отчетов (схем, рефератов) или направлены в базу знаний.

Работа может быть полезной в построении теории смысла текстов, что является предметом исследований, прежде всего в лингвистике, а также в области математической логики.

1. МЕТОДЫ ПРЕДСТАВЛЕНИЯ СМЫСЛА ТЕКСТА НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Лексические функции, предложенные Мельчуком [1], можно представить на синтаксическом уровне в виде предикатов следующим образом. Если рассмотреть совокупность словоформ в языке, возникающих при склонениях существительных, спряжениях глаголов и т.д. (т.е. словарь), и считать, что x и y — слова или словосочетания из этой совокупности, то получаем предикаты типа:

$Syn(x, y)$, x, y — синонимы;

$S_i(x, y),\ i = 1,...,4$, y — типовое название i-го актанта для x;

$Destr\ (x, y)$, y — типовое название «агрессивного» действия (x = «оса», y = «жалит»);

$Doc\ (x, y)$, y — «документ»: $Doc_{res}(x, y)$, y — «документ», являющийся результатом («воплощающий в себе»; x = «отчитываться», y = «отчет»), $Doc_{perm}(x, y)$,

y — «документ на право…» (x = «поезд», y = «(проездной) билет»), $Doc_{cert}(x, y)$,

y — «документ, удостоверяющий…» (x = «высшее образование», y = «диплом»). Для последних можем записать формулу:

$\forall x \forall y (Doc\ (x, y) \leftrightarrow Doc_{res}(x, y) \vee Doc_{perm}(x, y) \vee Doc_{cert}(x, y))$ и т.д.

В дальнейшем планируется рассматривать различные модели соответствующих сигнатур, в которых будут истинны данные предикаты.

Теоретико-множественные модели языков Маркуса [3] строятся следующим образом. Рассматривается некоторое разбиение словаря (он считается конечным множеством) естественного языка на классы. С помощью такого разбиения можно дать формальное определение грамматического рода или категории падежа. Маркус также вводит понятие синтаксических типов, которые приблизительно соответствуют традиционным частям речи.

Осуществляя операции над синтаксическими типами, можно определить грамматическую правильность предложения на естественном языке.

2. ГРАММАТИЧЕСКИЕ ПРЕДИКАТЫ

В данном разделе вводятся различные грамматические предикаты и описываются их свойства посредством логических формул. По сути дела, излагаемый материал является формализацией различных лингвистических знаний из работ [5, 6, 7].

1. $N(x,y_1,...,y_n)$, x — существительное, y_i — признаки, по которым существительные разделяем на группы. Будем считать, что запись $N(x,y_1,...,\underset{i}{0},...,y_n)$ обозначает отсутствие i-го признака. Если признаки $y_1,...,y_n$ взаимоисключающие, т.е. данное слово не может иметь сразу несколько или все признаки, то будем это обозначать $N(x,y)$, где $y = y_i$, если x имеет признак y_i. Например, существительное не может быть одновременно и в единственном, и во множественном числе. Но существительное может одновременно стоять в разных падежах (*метро*), иметь мужской и женский род (*плакса*), являться одушевлённым и неодушевлённым (*пень*). Из вышесказанного напрямую следует, что $y, y_1,..., y_n$ — константы, заранее заданные и в каждом случае специально оговоренные.

В связи с этим операция «исключающего или» определяется по-разному. Для предикатов типа $P(x,y_1,...,y_n)$ она определяется как конъюнкция дизъюнкций (например, см. самую первую формулу), а для предикатов типа $P(x,y)$ эта операция совпадает с обычной «или» (например, см. формулу из раздела ***существительные*** п. 5).

Для остальных частей речи эти обозначения аналогичны.

2. $Adj(x,y_1,...,y_n)$, x — прилагательное, y_i — признаки, по которым прилагательные разделяем на группы.

3. $Num(x,y_1,...,y_n)$, x — числительное, y_i — признаки, по которым числительные разделяем на группы.

4. $ProN(x,y_1,...,y_n)$, x — местоимение, y_i — признаки, по которым местоимения разделяем на группы.

5. $V(x,y_1,...,y_n)$, x — глагол, y_i — признаки, по которым глаголы разделяем на группы.

6. $PartP(x,y_1,...,y_n)$, x — причастие, y_i — признаки, по которым причастия разделяем на группы.

7. *VA(x,y1,...,yn), x* — деепричастие, y_i — признаки, по которым деепричастия разделяем на группы.

8. *AdV(x,y1,...,yn), x* — наречие, y_i — признаки, по которым наречия разделяем на группы.

9. *Prep(x,y1,...,yn), x* — предлог, y_i — признаки, по которым предлоги разделяем на группы.

10. *Con(x,y1,...,yn), x* — союз, y_i — признаки, по которым союзы разделяем на группы.

11. *PartL(x,y1,...,yn), x* — частица, y_i — признаки, по которым частицы разделяем на группы.

Нижний индекс соответствует порядковому номеру выделенного признака, верхний — номеру свойства, конкретизирующего этот признак.

Введём ещё один предикат $Plur(x)$. Он истинен, если x — существительное, употребляющееся исключительно во множественном числе (*Альпы, Мытищи, консервы, ворота, ножницы* и др.).

2.1. Имя существительное

1. $N_1(x, y)$, где y = «нар», если x — нарицательное существительное и y = «собств», если x — собственное существительное.

Нарицательные имена существительные представляют собой обобщённые названия однородных предметов:

а) $N_1^1(x)$ — лица (*мать, слесарь, девушка* и др.);

б) $N_1^2(x)$ — животные и птицы (*лошадь, жираф, соловей* и др.);

в) $N_1^3(x)$ — явления природы (*заморозки, град, гром* и др.);

г) $N_1^4(x)$ — явленияобщественной жизни (*демонстрация, выборы, революция* и др.);

д) $N_1^5(x)$ — отвлечённые понятия (*восприятие, глубина, демократия* и др.);

е) $N_1^6(x)$ — предметы (*газета, пылесос, часы* и др.);

ж) $N_1^7(x)$ — действия (*бег, чистка, состязание* и др.).

$$(\forall x)\left(N_1(x, нар) \leftrightarrow \mathop{\&}_{\substack{i=1\\ i\neq j}}^{7}\left(\neg\left(N_1^i(x)\to N_1^j(x)\right)\vee\neg\left(N_1^j(x)\to N_1^i(x)\right)\right)\right).$$

Нарицательное существительное — существительное, принадлежащее одной из групп а)—ж).

Эквивалентная запись этой формулы:

$$(\forall x)\left(N_1(x, нар) \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i\neq j}}^{7} \left(\left(N_1^i(x) \,\&\, \neg N_1^j(x) \right) \vee \left(N_1^j(x) \,\&\, \neg N_1^i(x) \right) \right) \right).$$

Собственные имена существительные являются названиями отдельных лиц, животных и единичных в своём роде предметов (пишутся с заглавной буквы):

а) $N_1^8(x)$ — имена, отчества и фамилии людей;

б) $N_1^9(x)$ — клички животных;

в) $N_1^{10}(x)$ — общественно-политические и исторические наименования, названия газет, журналов и литературных произведений;

г) $N_1^{11}(x)$ — географические наименования;

д) $N_1^{12}(x)$ — названия разновидностей сортов и марок (*конфеты «Птичье молоко»* и др.).

$(\forall x)\left(N_1(x, собств) \leftrightarrow \mathop{\&}_{\substack{i=8 \\ i\neq j}}^{12} \left(\neg\left(N_1^i(x) \to N_1^j(x) \right) \vee \neg\left(N_1^j(x) \to N_1^i(x) \right) \right) \right)$ — собственное существительное — существительное, относящееся к одной из групп а)—д).

Эквивалентная запись этой формулы:

$$(\forall x)\left(N_1(x, собств) \leftrightarrow \mathop{\&}_{\substack{i=8 \\ i\neq j}}^{12} \left(\left(N_1^i(x) \,\&\, \neg N_1^j(x) \right) \vee \left(N_1^j(x) \,\&\, \neg N_1^i(x) \right) \right) \right).$$

2. $N_2(x, y_1, y_2)$, y_1 — одушевлённое, y_2 — неодушевлённое. Заметим, что данные сведения есть в морфологическом словаре Dialing.

$(\forall x)\left(N_2(x, y_1, 0) \to \neg N_2(x, 0, y_2) \right)$ — формула исключения.

В обратную сторону импликации нет, т.к. неодушевлённые существительные могут переходить в одушевлённые, употребляясь в переносном смысле (*Попробуй убедить этого* ***пня***). Поэтому для таких существительных истинен предикат $N_2(x, y_1, y_2)$.

3. $N_3(x)$ — собирательные (collective noun) — слова, обозначающие совокупность живых существ или предметов в виде неделимого целого. Бы-

вают двух видов: собирательное значение выражено только основой (*знать, посуда* и др.) и не только основой, но и суффиксами ***-ество*** *(человечество* и др.*)*, ***-je*** *(тряпьё* и др.*)*, ***-ат*** *(старостат* и др.*)*, ***-ня*** *(родня* и др.*)*, ***-ура*** *(агентура* и др.*)*, ***-ија*** *(гвардия* и др.*)*, ***-ота*** *(пехота* и др.*)*, ***-итет*** *(генералитет* и др.*)*, ***-ара*** *(мошкара* и др.*)*.

Любое собирательное существительное должно иметь соотносительное существительное, обозначающее единичный предмет (*пехота — пехотинец*). Поэтому сюда не относятся слова типа *лес, толпа, народ, куча* и т.д.

4. $N_4(x)$ — вещественные — группа нарицательных существительных, слова которой употребляются для обозначения однородных по составу веществ, которые можно делить на части, обладающие свойствами целого, подвергать измерению, но не считать. Это названия пищевых продуктов, химических элементов и их соединений, сельскохозяйственных культур, полезных ископаемых, различных материалов (*мука, молоко, хлеб, консервы, натрий, спирт, пшеница, виноград, кожа, кирпич* и др.).

$(\forall x)\left(N_4(x) \to N_1(x, нар)\right)$ — вещественные существительные являются нарицательными;

$(\forall x)\left(N_4(x) \to N_2(x, 0, y_2)\right)$ — вещественные существительные являются неодушевлёнными.

5. $N_5(x, y)$, y = «отвл», если существительное отвлечённое, y = «конкр», если существительное конкретное.

Отвлечённые существительные образуют группу слов, которые обозначают различные абстрактные понятия (*упругость, белизна, хромота, синева, сострадание, именины, социология* и др.). Конкретные существительные обозначают отдельные предметы, живые существа и отдельные явления окружающей действительности (*костюм, термометр, живописец, вечер, наводнение, революция* и др.).

$(\forall x)\left(N_1(x, собст) \to \neg\left(N_5(x, отвл) \vee N_5(x, конкр)\right)\right)$ или

$(\forall x)\left(\left(N_5(x, отвл) \vee N_5(x, конкр)\right) \to N_1(x, нар)\right)$ — формулы, обозначающие, что отвлечённое или конкретное существительное является нарицательным. Можно переписать через конъюнкцию.

6. $N_6(x, y)$ — категория числа: y = «ед», если x — существительное в единственном числе, y = «мн», если x — существительное во множествен-

ном числе. Число существительных можно определить в морфологическом словаре Dialing.

7. $N_7(x, y_1, y_2, y_3)$ — показатель рода существительных: y_1 — мужской, y_2 — женский, y_3 — средний. Есть в морфологическом словаре Dialing.

Существительные общего рода — слова, с помощью которых в языке называются лица, отличающиеся друг от друга только полом (мужской и женский отличаются по контексту). Такие существительные изменяют свой род в зависимости от пола лица, с которым согласуется существительное (несклоняемые фамилии, *Саша, забияка, плакса; врач* имеет значение м.р.). По умолчанию ставим и мужской род, и женский, т.е. для них истинен предикат $N_7(x, y_1, y_2, 0)$.

$(\forall x)\Big(\big(N_6(x, ед) \vee N_6(x, мн)\big) \leftrightarrow \big(\neg\big(\big(N_7(x, y_1, 0, 0) \vee N_7(x, 0, y_2, 0)\big) \rightarrow N_7(x, 0, 0, y_3)\big) \vee$
$\vee \neg\big(N_7(x, 0, 0, y_3) \rightarrow \big(N_7(x, y_1, 0, 0) \vee N_7(x, 0, y_2, 0)\big)\big)\big)\Big)$ — существительные в единственном или во множественном числе могут быть одного из родов: мужского, женского или среднего рода; или мужского и женского одновременно — других возможностей нет.

$(\forall x)\Big(Plur(x) \leftrightarrow \big(\neg N_7(x, y_1, 0, 0) \,\&\, \neg N_7(x, 0, y_2, 0) \,\&\, \neg N_7(x, 0, 0, y_3)\big)\Big)$ — существительные, употребляющиеся исключительно во множественном числе, рода не имеют.

$(\forall x)\Big(Plur(x) \rightarrow \neg\big(N_6(x, ед)\big)\Big)$ — существительное, употребляющееся исключительно во множественном числе, естественно, не имеет единственного числа.

8. $N_8(x, y_1, y_2, y_3, y_4, y_5, y_6)$ — категория падежа: y_1 — именительный, y_2 — родительный, y_3 — дательный, y_4 — винительный, y_5 — творительный, y_6 — предложный. Есть в морфологическом словаре Dialing.

Словообразование имён существительных ([5]).

9. $N_9(x)$ — существительные, образованные префиксальным способом:

а) $N_9^1(x)$ — производные с общим значением интенсивности, высокой степени того, что названо производящей основой;

б) $N_9^2(x)$ — производные с общим значением противоположности, отрицания;

в) $N_9^3(x)$ — производные с общим значением неистинности, ложности;

г) $N_9^4(x)$ — производные со значением совместности;

д) $N_9^5(x)$ — производные со значением подчиненности.

10. $N_{10}(x)$ — существительные, образованные префиксально-суффиксальным способом:

а) $N_{10}^1(x)$ — производные с пространственными значениями;

б) $N_{10}^2(x)$ — производные с временными значениями;

в) $N_{10}^3(x)$ — производные, обозначающие отсутствие того, что названо производящей основой.

2.2. Имя прилагательное

1. $Adj_1(x, y)$ — деление прилагательных по разрядам: y = «кач», если прилагательное качественное, y = «отнс», если относительное, y = «прит», если притяжательное.

Качественные прилагательные могут обозначать:

а) $Adj_1^1(x)$ — цвет (*жёлтый, синий* и др.);

б) $Adj_1^2(x)$ — пространственные отношения (*длинный, прямой* и др.);

в) $Adj_1^3(x)$ — временные отношения (*долгий, быстрый* и др.);

г) $Adj_1^4(x)$ — свойства и качества вещей, воспринимаемые органами чувств (*сладкий, горячий* и др.);

д) $Adj_1^5(x)$ — физические качества людей и животных (*сильный, слепой* и др.);

е) $Adj_1^6(x)$ — духовные качества людей (*добрый, умный* и др.).

$$(\forall x)\left(Adj_1(x, кач) \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i\neq j}}^{6} \left(\neg\left(Adj_1^i(x) \to Adj_1^j(x) \right) \vee \neg\left(Adj_1^j(x) \to Adj_1^i(x) \right) \right) \right)$$ —

качественные прилагательные — прилагательные, принадлежащие одной из групп а)—е).

Эквивалентная запись этой формулы:

$$(\forall x)\left(Adj_1(x, кач) \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i\neq j}}^{6} \left(\left(Adj_1^i(x) \,\&\, \neg Adj_1^j(x) \right) \vee \left(Adj_1^j(x) \,\&\, \neg Adj_1^i(x) \right) \right) \right).$$

Относительные прилагательные обозначают признаки не непосредственно, а через отношение:

а) $Adj_1^7(x)$ — к материалу, из которого что-то сделано (*железный, молочный* и др.);

б) $Adj_1^8(x)$ — к месту (*московский, здешний* и др.);

в) $Adj_1^9(x)$ — ко времени (*вчерашний, летний* и др.);

г) $Adj_1^{10}(x)$ — к лицу (*детский, студенческий* и др.);

д) $Adj_1^{11}(x)$ — к понятию (*философский, научный* и др.);

е) $Adj_1^{12}(x)$ — к действию (*стиральный, подготовительный* и др.);

ж) $Adj_1^{13}(x)$ — к числу (*двойной, тройной* и др.);

$(\forall x)\left(Adj_1(x, отнс) \leftrightarrow \underset{\substack{i=7 \\ i\neq j}}{\overset{13}{\&}} \left(\neg\left(Adj_1^i(x) \to Adj_1^j(x) \right) \vee \neg\left(Adj_1^j(x) \to Adj_1^i(x) \right) \right) \right)$ — относительные прилагательные — прилагательные, принадлежащие одной из групп а)—ж).

Эквивалентная запись этой формулы:

$$(\forall x)\left(Adj_1(x, отнс) \leftrightarrow \underset{\substack{i=7 \\ i\neq j}}{\overset{13}{\&}} \left(\left(Adj_1^i(x) \,\&\, \neg Adj_1^j(x) \right) \vee \left(Adj_1^j(x) \,\&\, \neg Adj_1^i(x) \right) \right) \right).$$

$(\forall x)\left(Adj_1(x, отнс) \to \neg(Adj_5(x, сравн) \vee Adj_5(x, прев)) \right)$ — относительные прилагательные не изменяются по степеням сравнения;

$(\forall x)\left(Adj_1(x, отнс) \to \neg\left(Adj_{10}(x, полн) \vee Adj_{10}(x, кр) \right) \right)$ — относительные прилагательные не образуют кратких форм;

$(\forall x)\left(Adj_1(x, отнс) \to \neg Adj_8(x) \right)$ — относительные прилагательные не образуют форм субъективной оценки. Импликация только в одну сторону, т.к. прилагательное, обладая вышеперечисленными свойствами, может принадлежать к разряду притяжательных.

Притяжательные прилагательные обозначают принадлежность предмета лицу или животному и образуются с помощью суффиксов ***-ов (-ев-)*** (*дедов кабинет* и др.)***, -ин- (-ын-)*** (*гусиная лапка* и др.), ***-ач- (-яч-)*** (*мышиная нора* и др.) и ***-ий (-ja-, -je-)*** (*собачья конура* и др.).

$(\forall x)\left(Adj_1(x, прит) \to \neg(Adj_5(x, сравн) \vee Adj_5(x, прев))\right)$ — притяжательные прилагательные не изменяются по степеням сравнения;

$(\forall x)\left(Adj_1(x, прит) \to \neg\left(Adj_{10}(x, полн) \vee Adj_{10}(x, кр)\right)\right)$ — притяжательные прилагательные не образуют кратких форм;

$(\forall x)\left(Adj_1(x, прит) \to \neg Adj_8(x)\right)$ — притяжательные прилагательные не образуют форм субъективной оценки. Импликация только в одну сторону, так как прилагательное, обладая вышеперечисленными свойствами, может принадлежать к разряду относительных.

2. $Adj_2(x, y)$ — категория числа прилагательного: y = «ед», если x — прилагательное в единственном числе; y = «мн», если x — прилагательное во множественном числе.

3. $Adj_3(x, y)$ — категория рода прилагательных: y = «мр», если x — прилагательное мужского рода; y = «жр», если x — прилагательное женского рода; y = «ср», если x — прилагательное среднего рода.

$(\forall x)\left(Adj_2(x, мн)\right) \leftrightarrow \left(\neg Adj_3(x, мр) \,\&\, \neg Adj_3(x, жр) \,\&\, \neg Adj_3(x, ср)\right)$ — когда прилагательное во множественном числе, то нельзя определить род.

То же самое означает формула:

$$(\forall x)\left(Adj_2(x, мн)\right) \leftrightarrow \neg\left(Adj_3(x, мр) \vee Adj_3(x, жр) \vee Adj_3(x, ср)\right).$$

$(\forall x)\left(Adj_2(x, ед) \leftrightarrow \left(Adj_3(x, мр) \vee Adj_3(x, жр) \vee Adj_3(x, ср)\right)\right)$ — если прилагательное в единственном числе, то оно обязательно либо мужского рода, либо женского, либо среднего и наоборот. Эту формулу можно переписать в эквивалентном виде с импликацией.

4. $Adj_4(x, y_1, y_2, y_3, y_4, y_5, y_6)$ — категория падежа прилагательного: y_1 — именительный, y_2 — родительный, y_3 — дательный, y_4 — винительный, y_5 — творительный, y_6 — предложный.

5. $Adj_5(x, y)$ — степени сравнения, где y = «сравн», если прилагательное в сравнительной степени; y = «прев», если прилагательное в превосходной степени.

Для сравнительной степени существует две формы выражения $Adj_6(x, y)$, где

а) y = «синт», если форма выражения сравнительной степени синтетическая (простая). Синтетическая форма определяется в морфологическом словаре Dialing;

б) y = «анлт», если форма выражения сравнительной степени аналитическая (сложная), т.е. представляет собой сочетание слова *более* с исходной формой прилагательного (*более широкий, более красивая* и др.).

$(\forall x)\left(Adj_5(x, сравн) \leftrightarrow \left(Adj_6(x, синт) \vee Adj_6(x, анлт)\right)\right)$ — если прилагательное в сравнительной степени, то обязательно в одной из двух форм и, наоборот.

Превосходная степень прилагательных имеет три формы выражения $Adj_7(x, y)$, где

а) y = «синт», если форма выражения превосходной степени прилагательного синтетическая, т.е. образуется от основы исходной формы при помощи суффиксов ***-ейш-, -айш-*** (*высокий — высочайший* и др.);

б) y = «анлт», если форма выражения превосходной степени прилагательного аналитическая, т.е. образуется с помощью слова *самый* и исходной формы прилагательного (*самый строгий* и др.);

в) y = «сложн», если форма выражения превосходной степени прилагательного сложная, т.е. может образовываться тремя способами:

$Adj_7^1(x)$ — сочетание слова *наиболее* и исходной формы прилагательного (*наиболее честный* и др.);

$Adj_7^2(x)$ — сочетание формы сравнительной степени и слова *всех* или *всего* (*лучше всего* и др.);

$Adj_7^3(x)$ — сочетание отрицания *нет* и прилагательного в сравнительной степени (*нет красивее его* и др.).

$$(\forall x)\left(Adj_7(x, сложн) \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i \neq j}}^{3}\left(\neg\left(Adj_7^i(x) \to Adj_7^j(x)\right) \vee \neg\left(Adj_7^j(x) \to Adj_7^i(x)\right)\right)\right)$$ —

если в), то одна из последних трёх, и обратно. Для этой формулы существует эквивалентная запись без импликации.

$(\forall x)\left(Adj_5(x, прев) \leftrightarrow \left(Adj_7(x, синт) \vee Adj_7(x, анлт) \vee Adj_7(x, сложн)\right)\right)$ — прилагательное в превосходной степени ↔ обязательно в одной из трёх форм (аналогично формуле для сравнительной степени).

6. $Adj_8(x)$ — степени качества (формы субъективной оценки) обозначают степень проявления признака безотносительно к сравнению предметов. К степеням качества можно отнести следующие формы и словосочетания прилагательных:

а) $Adj_8^1(x)$ — приставочные формы прилагательных (*прехитрый, всесильный* и др.);

б) $Adj_8^2(x)$ — суффиксальные образования (*здоровенный, красноватый* и др.);

в) $Adj_8^3(x)$ — сочетания наречий меры и степени с исходной формой прилагательного (*весьма красивый, очень добрый* и др.);

г) $Adj_8^4(x)$ — повторение исходной формы прилагательного с префиксом или без него (*белый-белый, милый-премилый* и др.).

Принадлежность прилагательного к одной из четырёх групп можно определить только по словообразованию, перечислив приставки и суффиксы, при помощи которых образуется степень качества.

$$(\forall x)\left(Adj_8(x) \leftrightarrow \underset{\substack{i=1\\ i\neq j}}{\overset{4}{\&}} \left(\neg\left(Adj_8^i(x) \to Adj_8^j(x) \right) \vee \neg\left(Adj_8^j(x) \to Adj_8^i(x) \right) \right) \right)$$ — а)—г) $\leftrightarrow$ степени качества.

7. $Adj_9(x)$ — субстантивированные прилагательные — прилагательные, частично или полностью перешедшие в существительные (*мостовая, мороженое, портной* и др.). По своему значению делятся на пять групп, т.е. служат названиями следующих понятий:

а) $Adj_9^1(x)$ — лица (*рабочий, учёный* и др.);

б) $Adj_9^2(x)$ — помещения (*прихожая, учительская, прачечная, булочная* и др.);

в) $Adj_9^3(x)$ — документы (*накладная, дарственная* и др.);

г) $Adj_9^4(x)$ — пища и напитки (*отбивная, заливное, шампанское* и др.);

д) $Adj_9^5(x)$ — отвлечённые понятия (*прошлое, вечное, приданое* и др.).

$$(\forall x)\left(Adj_9(x) \leftrightarrow \underset{\substack{i=1\\ i\neq j}}{\overset{5}{\&}} \left(\neg\left(Adj_9^i(x) \to Adj_9^j(x) \right) \vee \neg\left(Adj_9^j(x) \to Adj_9^i(x) \right) \right) \right)$$ — субстантивированные прилагательные — слова, принадлежащие одной из групп а)—д).

8. $Adj_{10}(x, y)$, где y = «полн», если x — прилагательное в полной форме и y = «кр», если x — прилагательное в краткой форме. В морфологическом

словаре Dialing содержится информация о том, является ли прилагательное кратким.

$(\forall x)\big(Adj_{10}(x,кр) \leftrightarrow \neg\big(Adj_4(x,y_1,0,0,0,0,0) \vee Adj_4(x,0,y_2,0,0,0,0) \vee$

$\vee Adj_4(x,0,0,y_3,0,0,0) \vee Adj_4(x,0,0,0,y_4,0,0) \vee$

$\vee Adj_4(x,0,0,0,0,y_5,0) \vee Adj_4(x,0,0,0,0,0,y_6)\big)\big)$ — у кратких прилагательных нет падежа; здесь простое «или», т.к. формы некоторых падежей могут совпадать.

Словообразование имён прилагательных [5]:

9. $Adj_{11}(x)$ — прилагательные, образованные префиксальным способом:

а) $Adj_{11}^1(x)$ — категория прилагательных, обозначающих интенсивность, полноту проявления признака;

б) $Adj_{11}^2(x)$ — прилагательные, в которых приставки имеют значение отрицания, противоположности.

10. $Adj_{12}(x)$ — прилагательные, образованные префиксально-суффиксальным способом:

а) $Adj_{12}^1(x)$ — прилагательные, производимые на базе сочетаний существительных с предлогами, преобразуемыми в составе производных в приставки, могут включать разнообразные приставки, а из суффиксов *-н-* (преимущественно), реже *-ов-, -ск-* и некоторые другие;

б) $Adj_{12}^2(x)$ — на базе соединения отрицания *не* с сочетанием имени существительного с предлогом *без* образуются имена прилагательные со сложной приставкой *небез-* и суффиксом *-н-*. Производное имеет значение неполноты, слабой степени проявления признака;

в) $Adj_{12}^3(x)$ — прилагательные, образованные префиксально-суффиксальным способом от основ глаголов, образуются с помощью приставки *не-* и суффиксов *-н-* и *-м-*. Производные обозначают «невозможность подвергнуться действию».

2.3. Имя числительное

1. $Num_1(x,y)$ — деление числительных по синтаксическому употреблению: y = «колич», если x — количественное числительное; y = «собир», если x — собирательное числительное (*оба, пятеро* и др.); y = «пор», если x —

порядковое числительное (*пятнадцатый, первые* и др.). Порядковые числительные определяются из словаря Dialing.

Количественные числительные делятся на

а) $Num_1^1(x)$ — собственно-количественные (*один* и др.),

б) $Num_1^2(x)$ — дробные (*две пятых* и др.),

в) $Num_1^3(x)$ — неопределённо-количественные (*много* и др.).

$$(\forall x)\left(Num_1(x, колич) \leftrightarrow \underset{\substack{i=1 \\ i\neq j}}{\overset{3}{\&}} \left(\neg\left(Num_1^i(x) \to Num_1^j(x) \right) \vee \neg\left(Num_1^j(x) \to Num_1^i(x) \right) \right) \right) -$$

колич. ↔ а)—в).

2. $Num_2(x, y)$ — деление числительных по структуре, где y = «прост», если x — простое числительное, т.е. состоит из одного корня (*четыре* и др.); y = «слож», если x — сложное числительное, т.е. имеет две основы (образованы из простых) (*пятьдесят* и др.); y = «сост», если x — составное числительное, т.е. образовано сочетанием простых или сложных числительных (*двести одиннадцать* и др.), сюда же относятся дробные числительные.

$(\forall x)\left(Num_1^2(x) \to Num_2(x, сост) \right)$ — верна импликация дробные → составные.

3. $Num_3(x, y)$ — категория рода: y = «мр», если x — числительное мужского рода; y = «жр», если x — числительное женского рода, y = «ср», если x — числительное среднего рода.

$(\forall x)\left(\left(Num_1(x, пор) \,\&\, Num_4(x, ед) \right) \leftrightarrow \left(Num_3(x, мр) \vee Num_3(x, жр) \vee Num_3(x, ср) \right) \right) -$ род имеют только порядковые в единственном числе (ИСКЛЮЧЕНИЯ: *один - одна - одно, два - две, оба - обе, полтора - полторы* и *тысяча, миллион* и т.п.).

$(\forall x)\left(Num_1(x, колич) \to \neg\left(Num_3(x, мр) \vee Num_3(x, жр) \vee Num_3(x, ср) \right) \right)$ — количественные числительные не изменяются по родам (ИСКЛЮЧЕНИЯ: *один - одна - одно, два - две, оба - обе, полтора - полторы* и *тысяча, миллион* и т.п.).

$(\forall x)\left(Num_1(x, собир) \to \neg\left(Num_3(x, мр) \vee Num_3(x, жр) \vee Num_3(x, ср) \right) \right)$ — собирательные числительные не изменяются по родам

4. $Num_4(x, y)$ — категория числа: y = «ед», если x — числительное в единственном числе; y = «мн», если x — числительное во множественном числе.

$(\forall x)\left(Num_1(x, пор) \leftrightarrow \left(Num_4(x, ед) \vee Num_4(x, мн)\right)\right)$ — число есть только для порядковых числительных (ИСКЛЮЧЕНИЯ: количественные числительные *один, тысяча* и т.п.).

$(\forall x)\left(Num_1(x, колич) \rightarrow \neg\left(Num_4(x, ед) \vee Num_4(x, мн)\right)\right)$ — количественные числительные не изменяются по числам (ИСКЛЮЧЕНИЯ: количественные числительные *один, тысяча* и т.п.).

$(\forall x)\left(Num_1(x, собир) \rightarrow \neg\left(Num_4(x, ед) \vee Num_4(x, мн)\right)\right)$ — собирательные числительные не изменяются по числам.

5. $Num_5(x, y_1, y_2, y_3, y_4, y_5, y_6)$ — категория падежа: y_1 — именительный, y_2 — родительный, y_3 — дательный, y_4 — винительный, y_5 — творительный, y_6 — предложный.

2.4. Местоимение

1. $ProN_1(x, y)$ — разряды местоимений по значению:
   - y = «лич», если x — личное (*я, вы* и др.);
   - y = «возвр», если x — возвратное (*себя*);
   - y = «прит», если x — притяжательное (*свой, ваш* и др.). Они определяются из словаря Dialing (правда, почему-то к ним относят местоимения ещё из некоторых других разрядов).
   - y = «указ», если x — указательное (*этот, такой* и др.);
   - y = «вопр», если x — вопросительное (*кто, который, сколько* и др.);
   - y = «отн», если x — относительное, т.е. вопросительное, но используемое для связи частей сложного предложения (*кто, который* и др.);
   - y = «опр», если x — определительное (*сам, самый, весь, всякий* и др.);
   - y = «отр», если x — отрицательное (*никто, некого, никакой, ничей* и др.);
   - y = «неопр», если x — неопределённое (*некто, несколько, кое-кто, что-то* и др.).

2. $ProN_2(x, y)$ — категория лица: y = «1 л», если x — местоимение первого лица (*я, мы*); y = «2 л», если x — местоимение второго лица (*ты, вы*); y = «3 л», если x — местоимение третьего лица (*он, она, оно, они*).

$(\forall x)\left(ProN_1(x, лич) \to \left(ProN_2(x, 1л) \vee ProN_2(x, 2л) \vee ProN_2(x, 3л)\right)\right)$ — категория лица определена для личных местоимений.

$(\forall x)\left(ProN_1(x, прит) \to \left(ProN_2(x, 2л) \vee ProN_2(x, 3л)\right)\right)$ — категория лица определена для притяжательных местоимений (у них есть только 2-ое и 3-е).

$(\forall x)\Big(\big(ProN_1(x, возвр) \vee ProN_1(x, указ) \vee ProN_1(x, вопр) \vee ProN_1(x, отн) \vee$

$\vee ProN_1(x, отр) \vee ProN_1(x, опр) \vee ProN_1(x, неопр)\big) \to \neg\big(ProN_2(x, 1л) \vee$

$\vee ProN_2(x, 2л) \vee ProN_2(x, 3л)\big)\Big)$ — для остальных категория лица не определена.

3. $ProN_3(x, y)$ — категория рода: y = «мр», если x — местоимение мужского рода, y = «жр», если x — местоимение женского рода, y = «ср», если x — местоимение среднего рода.

$(\forall x)\Big(\big(ProN_6(x, мсущ) \vee ProN_6(x, мприл)\big) \leftrightarrow$

$\leftrightarrow \left(ProN_3(x, мр) \vee ProN_3(x, жр) \vee ProN_3(x, ср)\right)\Big)$ — определена для местоимений-существительных и местоимений-прилагательных.

$(\forall x)\left(ProN_6(x, мчисл) \leftrightarrow \neg\left(ProN_3(x, мр) \vee ProN_3(x, жр) \vee ProN_3(x, ср)\right)\right)$ — не определена для местоимений-числительных.

$(\forall x)\Big(\big(\left(ProN_1(x, лич)\ \&\ ProN_2(x, 3л)\right) \vee ProN_1(x, указ) \vee ProN_1(x, прит) \vee$

$\vee ProN_1(x, опр) \vee ProN_1(x, отн) \vee ProN_1(x, вопр)\big) \leftrightarrow$

$\leftrightarrow \left(ProN_3(x, мр) \vee ProN_3(x, жр) \vee ProN_3(x, ср)\right)\Big)$ — определена для личных местоимений 3-его лица, а также для указательных, вопросительных, относительных, определительных, притяжательных. В формулу не включены отрицательные и неопределённые местоимения, т.к. НЕ ДЛЯ ВСЕХ местоимений из этих групп определена категория рода.

$(\forall x)\left(ProN_1(x, возвр) \to \neg\left(ProN_3(x, мр) \vee ProN_3(x, жр) \vee ProN_3(x, ср)\right)\right)$ — не определена для возвратного местоимения.

4. $ProN_4(x, y)$ — категория числа: y = «ед», если x — местоимение в единственном числе; y = «мн», если x — местоимение во множественном числе.

$(\forall x)\left(ProN_1(x, возвр) \to \neg\left(ProN_4(x, мн) \vee ProN_4(x, ед)\right)\right)$ — определена для всех, кроме возвратного *себя.*

$(\forall x)\left(\left(ProN_6(x, мсущ) \vee ProN_6(x, мприл)\right) \leftrightarrow \left(ProN_4(x, мн) \vee ProN_4(x, ед)\right)\right)$ – определена для местоимения-существительного и местоимения-прилагательного.

$(\forall x)\left(ProN_6(x, мчисл) \leftrightarrow \neg\left(ProN_4(x, мн) \vee ProN_4(x, ед)\right)\right)$ — не определена для местоимения-числительного.

5. $ProN_5(x, y_1, y_2, y_3, y_4, y_5, y_6)$ — категория падежа: y_1 — именительный, y_2 — родительный, y_3 — дательный, y_4 — винительный, y_5 — творительный, y_6 — предложный.

$(\forall x)\Big(\left(ProN_6(x, мсущ) \vee ProN_6(x, мприл)\right) \leftrightarrow \left(ProN_5(x, y_1, 0, 0, 0, 0, 0) \vee \right.$

$\vee ProN_5(x, 0, y_2, 0, 0, 0, 0) \vee ProN_5(x, 0, 0, y_3, 0, 0, 0) \vee ProN_5(x, 0, 0, 0, y_4, 0, 0) \vee$

$\left.\vee ProN_5(x, 0, 0, 0, 0, y_5, 0) \vee ProN_5(x, 0, 0, 0, 0, 0, y_6)\right)\Big)$ — категория падежа определена для местоимения-существительного и местоимения-прилагательного, не определена для местоимения-числительного.

6. $ProN_6(x, y)$ — разряды местоимений в зависимости от соотнесённости их с другими частями речи, т.е. y = «мсущ», если x — местоимённое существительное (*ты, себя, никто, некто, что-то* и др.); y = «мприл», если x — местоимённое прилагательное (*мой, этот, сам, всякий, такой* и др.); y = «мчисл», если x — местоимённое числительное (*сколько, несколько, столько* и др.).

2.5. Глагол

1. $V_1(x, y)$ — категория времени, где y = «нст», если x — глагол в настоящем времени, y = «прш», если x — глагол в прошедшем времени, y = «буд», если x — глагол в будущем времени. Будущее время глагола имеет две следующие формы:

а) $V_1^1(x)$ — простое будущее (*прочитаю*). Настоящее, прошедшее и простое будущее времена определяются из морфологического словаря Dialing;

б) $V_1^2(x)$ — сложное будущее (*буду читать*). Образуется из спрягаемых форм глагола *быть* и инфинитива основного глагола.

$(\forall x)\left(V_1(x,буд) \leftrightarrow \left(\left(V_1^1(x) \,\&\, \neg V_1^2(x)\right) \vee \left(V_1^2(x) \,\&\, \neg V_1^1(x)\right)\right)\right)$ — глагол в будущем времени ↔ а) или б).

$(\forall x)\left(V_1(x,нст) \rightarrow \left(V_2(x,y_1,0) \vee V_2(x,y_1,y_2)\right)\right)$ — настоящее время возможно только для глаголов несовершенного вида или двухвидовых.

$(\forall x)\left(V_1(x,нст) \rightarrow \neg V_2(x,0,y_2)\right)$ — то же самое, только переформулировано: настоящее время не существует для глаголов совершенного вида.

$(\forall x)\left(V_1^1(x) \rightarrow \left(V_2(x,0,y_2) \vee V_2(x,y_1,y_2)\right)\right)$ — простое будущее время существует только для глаголов совершенного вида или двухвидовых.

$(\forall x)\left(V_1^1(x) \rightarrow \neg V_2(x,y_1,0)\right)$ — то же самое, только переформулировано: если глагол в форме простого будущего времени, то он несовершенного вида.

$(\forall x)\left(V_1^2(x) \rightarrow \left(V_2(x,y_1,0) \vee V_2(x,y_1,y_2)\right)\right)$ — сложное будущее время существует только для глаголов несовершенного вида или двухвидовых.

$(\forall x)\left(V_1^2(x) \rightarrow \neg V_2(x,0,y_2)\right)$ — то же самое, только переформулировано: если глагол в форме сложного будущего времени, то он несовершенного вида.

$(\forall x)\left(V_4(x,изъяв) \leftrightarrow \left(V_1(x,нст) \vee V_1(x,прш) \vee V_1(x,буд)\right)\right)$ — время можно определять только для глаголов в форме изъявительного наклонения.

$(\forall x)\left(\left(V_4(x,сосл) \vee V_4(x,пов)\right) \leftrightarrow \neg\left(V_1(x,нст) \vee V_1(x,прш) \vee V_1(x,буд)\right)\right)$ — категория времени не определена для глаголов в сослагательном и повелительном наклонениях.

2. $V_2(x,y_1,y_2)$ — категория вида, где y_1 — несовершенный вид (глаголы, которые обозначают действия длительные, неограниченные в своём развитии, которые происходили в прошлом, до момента речи); y_2 — совершенный вид (глаголы, которые обозначают либо недлительные, мгновенные

однократные действия, либо ограниченные в своей длительности, либо уже закончившиеся).

Принадлежность глагола совершенному или несовершенному видам возможно определить из морфологического словаря Dialing. Если глагол двухвидовый, т.е. имеет значение совершенного или несовершенного вида в зависимости от контекста (*исследовать, обещать, организовать* и др.), то предикат будет иметь вид $V_2(x, y_1, y_2)$.

$(\forall x)\big(\big(V_2(x, y_1, y_2) \,\&\, V_2(x, y_1, 0)\big) \to V_1(x, нст)\big)$,

$(\forall x)\big(\big(V_2(x, y_1, y_2) \,\&\, V_1(x, нст)\big) \to V_2(x, y_1, 0)\big)$,

$(\forall x)\big(\big(V_2(x, y_1, y_2) \,\&\, V_2(x, 0, y_2)\big) \to V_1(x, буд)\big)$,

$(\forall x)\big(\big(V_2(x, y_1, y_2) \,\&\, V_1(x, буд)\big) \to V_2(x, 0, y_2)\big)$ — четыре формулы, показывающие, возможные ситуации для двухвидовых глаголов (настоящее время, несовершенный вид) или (будущее время, совершенный вид).

3. $V_3(x, y)$ — категория лица, где y = «1 л», если x — глагол в форме первого лица (употребляется для обозначения действий говорящего); y = «2 л», если x — глагол в форме второго лица (обозначает действия собеседника); y = «3 л», если x — глагол в форме третьего лица (обозначает действия лица, не участвующего в речи, являющегося предметом речи).

Значение лица передаётся личными местоимениями. Формы лиц глаголов содержатся в морфологическом словаре Dialing.

$(\forall x)\big(\big(V_1(x, прш) \,\&\, \neg V_4(x, сосл)\big) \vee \big(V_4(x, сосл) \,\&\, \neg V_1(x, прш)\big) \leftrightarrow$

$\leftrightarrow \neg\big(V_3(x, 1л) \vee V_3(x, 2л) \vee V_3(x, 3л)\big)\big)$ — лицо нельзя определить для глаголов в прошедшем времени и для глаголов сослагательного наклонения.

4. $V_4(x, y)$ — категория наклонения, где y = «изъяв», если x — глагол в форме изъявительного (*читаю, читал, буду читать*); y = «сосл», если x — глагол в форме сослагательного наклонения (*читал бы*); y = «пов», если x — глагол в форме повелительного наклонения (*читай!*).

Повелительное наклонение определяется из словаря Dialing. Сослагательное наклонение образуется присоединением к прошедшему времени глагола частицы *бы*.

5. $V_5(x, y)$ — категория переходности или непереходности, где y = «нп», если глагол непереходный; y = «пе», если глагол переходный. Переходность/непереходность глаголов определяются в словаре Dialing.

Непереходные глаголы обозначают такие действия или состояния, которые не направлены на какой-либо объект.

Переходные глаголы обозначают действие, активно направленное на какой-либо объект. Для переходных глаголов определена категория залога $V_6(x, y)$:

а) y = «дст», если глагол действительного залога, т.е. относится к переходным глаголам (*Рабочий строит дом*);

б) y = «срвоз», если глагол средневозвратного залога, т.е. образован от переходного добавлением частицы ***-ся (-сь)***. В зависимости от лексического значения такие глаголы могут быть разделены на несколько групп:

$V_6^1(x)$ — глаголы с собственно-возвратным значением называют такое действие, производитель которого является одновременно и объектом действия (*мыться* и др.);

$V_6^2(x)$ — глаголы с общевозвратным значением указывает на внутреннее состояние субъекта, настроение, переживание, а также на внешние действия – движения, совершаемые субъектом (*успокаиваться* и др.);

$V_6^3(x)$ — глаголы с объектно-возвратным значением — обозначают такие действия, которые постоянно свойственны субъекту (*кошки царапаются*);

$V_6^4(x)$ — глаголы с взаимно-возвратным значением указывают на действие, которое совершается несколькими действующими лицами (*ссориться* и др.);

$V_6^5(x)$ — глаголы с косвенно-возвратным значением обозначают действие, совершаемое субъектом для себя, в своих интересах (*запасаться* и др.).

в) y = «страд», если глагол страдательного залога, т.е. образован от переходного добавлением частицы ***-ся (-сь)***, но, в отличие от глаголов средневозвратного залога, называет действие, которое испытывает на себе объект, подвергающийся действию (*Дом строится рабочими.*).

$(\forall x)\left(V_5(x, не) \leftrightarrow V_6(x, дст)\right)$ — формула взаимосвязи действительного залога и переходных глаголов (это одно и то же).

$(\forall, x)\left(V_6(x, \textit{срвоз}) \leftrightarrow \underset{\substack{i=1 \\ i \neq j}}{\overset{5}{\&}} \left(\left(V_6^i(x) \& \neg V_6^j(x)\right) \vee \left(V_6^j(x) \& \neg V_6^i(x)\right)\right)\right)$ — средневозвратный залог состоит из глаголов, принадлежащих одной из пяти групп.

6. $V_7(x, y)$ — категория рода: y = «мр», если x — глагол мужского рода, y = «жр», если x — глагол женского рода; y = «ср», если x — глагол среднего рода. Род глаголов определяется в словаре Dialing.

7. $V_8(x, y)$ — категория числа: y = «ед», если x — глагол в единственном числе; y = «мн», если x — глагол во множественном числе. В каком числе стоит глагол, определяется в словаре Dialing.

$(\forall x)\left(V_8(x, \textit{мн})\right) \leftrightarrow \neg\left(V_7(x, \textit{мр}) \vee V_7(x, \textit{жр}) \vee V_7(x, \textit{ср})\right)$ — когда глагол во множественном числе, то нельзя определить род.

$(\forall x)\left(\left(\left(\left(V_1(x, \textit{прш}) \& \neg V_4(x, \textit{сосл})\right) \vee \left(V_4(x, \textit{сосл}) \& \neg V_1(x, \textit{прш})\right)\right) \& V_8(x, \textit{ед})\right) \leftrightarrow$

$\leftrightarrow \left(V_7(x, \textit{мр}) \vee V_7(x, \textit{жр}) \vee V_7(x, \textit{ср})\right)\right)$ — если глагол в прошедшем времени в единственном числе или сослагательного наклонения в единственном числе, то он обязательно либо мужского рода, либо женского, либо среднего, и наоборот.

Эту формулу можно переписать в эквивалентном виде с импликацией в первой строке.

8. $V_9(x, y)$, y = «инф», если x — глагол в неопределённой форме (в форме инфинитива), y = «спрф», если x — глагол в спрягаемой форме (т.е. не инфинитив). Инфинитив глагола определяется в словаре Dialing.

$(\forall x)\left(V_9(x, \textit{инф}) \rightarrow \neg\left(V_7(x, \textit{мр}) \vee V_7(x, \textit{жр}) \vee V_7(x, \textit{ср})\right)\right)$ — у инфинитива нет рода.

$(\forall x)\left(V_9(x, \textit{инф}) \rightarrow \neg\left(V_8(x, \textit{ед}) \vee V_8(x, \textit{мн})\right)\right)$ — у инфинитива нет числа.

$(\forall x)\left(V_9(x, \textit{инф}) \rightarrow \neg\left(V_3(x, 1\textit{л}) \vee V_3(x, 2\textit{л}) \vee V_3(x, 3\textit{л})\right)\right)$ — у инфинитива нет лица.

Словообразование глаголов [5].

9. $V_{10}(x)$ — глаголы, образованные префиксальным способом:

а) $V_{10}^{1}(x)$ — глаголы с приставками пространственных значений, они обозначают различные направления действия;

б) $V_{10}^{2}(x)$ — глаголы, обозначающие начало процесса;

в) $V_{10}^{3}(x)$ — глаголы, обозначающие окончание процесса;

г) $V_{10}^{4}(x)$ — глаголы, обозначающие окончание действия с оттенками полноты, тщательности, энергичности, силы его выполнения;

д) $V_{10}^{6}(x)$ — глаголы, обозначающие полную исчерпанность предмета действием, а также причинение неприятности, ущерба действием;

е) $V_{10}^{7}(x)$ — глаголы, обозначающие дополнительное, добавочное действие, добавление чего-либо действием, а также слабость, неполноту действия.

10. $V_{11}(x)$ — глаголы, образованные префиксально-суффиксальным способом:

а) $V_{11}^{1}(x)$ — глаголы, обозначающие неполноту, ослабленность действия, имеющие прерывисто-длительное значение (схема образования таких глаголов: приставка + производящая основа + суффикс несовершенного вида);

б) $V_{11}^{2}(x)$ — глаголы, обозначающие интенсивность, тщательность совершения действия (схема образования таких глаголов: приставка + производящая основа + суффикс несовершенного вида);

в) $V_{11}^{3}(x)$ — глаголы, образуемые одновременным присоединение приставки и *-ся*, (схема образования таких глаголов: приставка + производящий глагол + *ся*);

г) $V_{11}^{4}(x)$ — глаголы, производимые от основ имён существительных префиксально-суффиксальным способом.

Для составного глагольного сказуемого вспомогательными могут выступать следующие глаголы [6]:

11. $V_{12}(x)$ — глаголы, обозначающие начало, конец, продолжение действия (*начать, стать, перестать, кончить продолжать* и др.).

12) $V_{13}(x)$ — глаголы с модальным значением, выражающие различные оттенки модальности:

а) $V_{13}^{1}(x)$ — возможность, невозможность, предрасположенность к действию, способность (*мочь, уметь, научиться, потрудиться* и др.);

б) $V_{13}^{2}(x)$ — желание, стремление, решение, старание (*хотеть, желать, намереваться, пытаться* и др.);

в) $V_{13}^{3}(x)$ — процессы мысли, психические переживания (*думать, затевать, надеяться, бояться, медлить, терпеть* и др.).

Для составного именного сказуемого глаголы-связки могут быть [6]:

13. $V_{14}(x)$ — с отвлечённым значением, т.е. они выполняют лишь грамматическую функцию и полностью лишены лексического значения (*быть, являться, есть* и *суть*);

14. $V_{15}(x)$ — полуотвлечённые (полузнаменательные), имеющие различные лексические значения:

а) $V_{15}^{1}(x)$ — проявления, обнаружения признака (*бывать, оказаться, оказываться* и др.),

б) $V_{15}^{2}(x)$ — признака в чьём-либо представлении (*казаться, представляться, считаться* и др.),

в) $V_{15}^{3}(x)$ — возникновение признака, перехода из одного состояния в другое или, наоборот, сохранения прежнего состояния (*стать, сделаться, остаться* и др.),

г) $V_{15}^{4}(x)$ — названия признака (*зваться, почитаться, называться*).

Эти глаголы-связки отличаются от соответствующих глаголов в прямом значении (*Сын **стал** взрослым. — Сын **стал** на колени.*).

15. $V_{16}(x)$ — знаменательные. К ним относятся глаголы с полным лексическим значением, обозначающие движение или состояние предмета (*жить, работать, сидеть, ходить, вернуться, родиться* и др.). Например, составное именное сказуемое есть в первом, а не во втором предложении: *Клоун **вышел** на улицу **одетый** в пальто. – Клоун **вышел** на улицу, одетый в пальто*.

2.6. Причастие

1. $PartP_1(x, y)$ — формы залога причастий, где y = «дст», если x — действительное причастие (называет признак действия, которое совершает или совершил сам субъект), y = «стр», если x — страдательное причастие (называет признак действия, которое испытывает или испытывал на себе носитель этого признака). Формы причастий определяются в словаре Dialing.

2. $PartP_2(x, y)$ — категория времени, где y = «нст», если x — причастие настоящего времени, y = «прш», если x — причастие прошедшего времени. Определяется в словаре Dialing.

3. $PartP_3(x, y)$ — переходность/непереходность, где y = «нп», если причастие образовано от непереходного глагола, y = «пе», если причастие образовано от переходного глагола. Переходность/непереходность причастий определяется в словаре Dialing.

4. $PartP_4(x, y)$ — вид, где y = «нсв», если x — причастие несовершенного вида, y = «св», если x — причастие совершенного вида. Определяется в словаре Dialing.

	наст. вр.	**прош. вр.**
действит.	несов. вид	любые
страдат.	несов. вид, перех.	перех.

Эту таблицу можно трактовать и по строкам и по столбцам.

Из таблицы получаем следующие формулы:

$(\forall x)\left(\left(PartP_1(x, стр) \,\&\, PartP_2(x, нст)\right) \leftrightarrow \left(PartP_3(x, пе) \,\&\, PartP_4(x, нсв)\right)\right)$ – страдательные причастия настоящего времени образуются только от переходных глаголов несовершенного вида.

$(\forall x)\left(\left(PartP_1(x, дст) \,\&\, PartP_2(x, нст)\right) \rightarrow PartP_4(x, нсв)\right)$ — действительные настоящего времени — от несовершенного вида.

$(\forall x)\left(\left(PartP_1(x, стр) \,\&\, PartP_2(x, прш)\right) \rightarrow PartP_3(x, пе)\right)$ — страдательные причастия прошедшего времени — от переходных глаголов.

$(\forall x)\left(PartP_4(x, св) \rightarrow PartP_2(x, прш)\right)$ — причастия совершенного вида бывают только прошедшего времени.

$(\forall x)\left(PartP_4(x, нсв) \rightarrow \left(PartP_2(x, прш) \vee PartP_2(x, нст)\right)\right)$ — причастия несовершенного вида бывают и настоящего, и прошедшего времени.

Последние две формулы можно записать иначе:

$(\forall x)\left(PartP_2(x, нст) \rightarrow PartP_4(x, нсв)\right)$ — причастия настоящего времени бывают только несовершенного вида;

$(\forall x)\left(PartP_2(x, прш) \to \left(PartP_4(x, св) \vee PartP_4(x, нсв)\right)\right)$ — причастия прошедшего времени бывают и совершенного, и несовершенного вида.

5. $PartP_5(x, y)$ — категория рода: y = «мр», если x — причастие мужского рода; y = «жр», если x — причастие женского рода; y = «ср», если x — причастие среднего рода. Определяется в словаре Dialing.

6. $PartP_6(x, y)$ — категория числа: y = «ед», если x — причастие в единственном числе; y = «мн», если x — причастие во множественном числе. Определяется в словаре Dialing.

$(\forall x)\left(PartP_6(x, мн)\right) \leftrightarrow \left(\neg PartP_5(x, мр) \,\&\, \neg PartP_5(x, жр) \,\&\, \neg PartP_5(x, ср)\right)$ — когда причастие во множественном числе, то нельзя определить род. То же самое означает формула:

$$(\forall x)\left(PartP_6(x, мн)\right) \leftrightarrow \neg\left(PartP_5(x, мр) \vee PartP_5(x, жр) \vee Part_5(x, ср)\right).$$

$(\forall x)\left(PartP_6(x, ед) \leftrightarrow \left(PartP_5(x, мр) \vee PartP_5(x, жр) \vee PartP_5(x, ср)\right)\right)$ — если причастие в единственном числе, то оно обязательно либо мужского рода, либо женского рода, либо среднего рода, и наоборот. Эту формулу можно переписать в эквивалентном виде с импликацией.

7) $PartP_7(x, y_1, y_2, y_3, y_4, y_5, y_6)$ — категория падежа: y_1 — именительный, y_2 — родительный, y_3 — дательный, y_4 — винительный, y_5 — творительный, y_6 — предложный. Определяется в словаре Dialing.

$(\forall x)\Big(PartP_8(x, кр) \leftrightarrow \neg\big(PartP_7(x, y_1, 0, 0, 0, 0, 0) \vee PartP_7(x, 0, y_2, 0, 0, 0, 0) \vee$

$\vee PartP_7(x, 0, 0, y_3, 0, 0, 0) \vee PartP_7(x, 0, 0, 0, y_4, 0, 0) \vee PartP_7(x, 0, 0, 0, 0, y_5, 0) \vee$

$\vee PartP_7(x, 0, 0, 0, 0, 0, y_6)\big)\Big)$ — для кратких причастий не определена категория падежа.

8) $PartP_8(x, y)$ — категория краткости/полноты: y = «полн», если x — причастие в полной форме и y = «кр», если x — причастие в краткой форме. В морфологическом словаре Dialing содержится информация о том, является ли причастие кратким.

$(\forall x)\left(PartP_8(x, кр) \to \left(PartP_1(x, стр) \,\&\, PartP_2(x, нст)\right) \vee \right.$

$\left. \vee \left(PartP_1(x, стр) \,\&\, PartP_2(x, прш)\right)\right)$ — краткие формы существуют только у страдательных причастий прошедшего времени и настоящего времени.

2.7. Деепричастие

1. $VA_1(x, y)$ — категория вида, где y = «нсв», если x — деепричастие несовершенного вида; y = «св», если x — деепричастие совершенного вида.

2.8. Наречие

1. $AdV_1(x, y_1, y_2)$ — разряды наречий, где y_1 означает, что наречие обстоятельственное, y_2 означает, что наречие определительное. Для наречий, являющихся обстоятельственными или определительными в зависимости от контекста, будем считать, что они принадлежат к двум разрядам одновременно, т.е. для них предикат имеет вид $AdV_1(x, y_1, y_2)$. К таким наречиям относится, например, наречие *прямо*. Оно может употребляться в значении «по прямой линии» и в значении «откровенно».

Обстоятельственные наречия обозначают различные условия (обстоятельства), в которых протекает действие:

а) $AdV_1^1(x)$ — время (*вчера, рано, иногда, сейчас* и др.);

б) $AdV_1^2(x)$ — место (*справа, назад, возле, там* и др.);

в) $AdV_1^3(x)$ — причина (*сгоряча, потому* и др.);

г) $AdV_1^4(x)$ — цель (*нарочно, насмех, в насмешку* и др.).

$$(\forall x)\left(\left(AdV_1(x, y_1, 0) \vee AdV_1(x, y_1, y_2)\right) \leftrightarrow \right.$$

$$\left. \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i \neq j}}^{4} \left(\neg\left(AdV_1^i(x) \to AdV_1^j(x)\right) \vee \neg\left(AdV_1^j(x) \to AdV_1^i(x)\right)\right)\right)$$ — наречие обстоятельственное или одновременно обстоятельственное и определительное — наречие, принадлежащее одной из групп а)—г).

Можно переписать эту формулу в эквивалентном виде без импликации.

Определительные (необстоятельственные) наречия $AdV_2(x, y_1, y_2, y_3)$ бывают трёх типов:

а) y_1 — качественные, т.е. выражают оценку действия (*хорошо, трусливо, иронически* и др.);

б) y_2 — количественные, т.е. указывают на меру (количество) действия или признака (*очень, гораздо, чересчур, почти* и др.). Существуют наречия,

которые в зависимости от контекста являются качественными или количественными. Например, наречие *легко: Шли легко раненные. — Он легко приподнялся*. В этом случае полагаем, что наречие относится одновременно к двум типам, т.е. $AdV_2(x, y_1, y_2, 0)$;

в) y_3 — способа и образа действия, т.е. обозначают, каким способом или образом протекает действие (*вброд, бегом, наизусть* и др.). Сюда также относятся наречия:

$AdV_2^1(x)$ — сравнения и уподобления (*по-дружески, по-прежнему* и др.);

$AdV_2^2(x)$ — совокупности (*толпой, вереницей, поодиночке, вшестером* и др.).

$(\forall x)\Big(\Big(\Big(AdV_2^1(x) \& \neg AdV_2^2(x)\Big) \vee \Big(AdV_2^2(x) \& \neg AdV_2^1(x)\Big)\Big) \to AdV_2(x, 0, 0, y_3)\Big)$ — если последних два → относятся к в), обратное неверно.

$$(\forall x)\Big(\big(AdV_1(x, 0, y_2) \vee AdV_1(x, y_1, y_2)\big) \leftrightarrow$$

$$\leftrightarrow \Big(\neg\Big(\big(AdV_2(x, y_1, 0, 0) \vee AdV_2(x, 0, y_2, 0)\big) \to AdV_2(x, 0, 0, y_3)\Big) \vee$$

$$\vee \neg\Big(AdV_2(x, 0, 0, y_3) \to \big(AdV_2(x, y_1, 0, 0) \vee AdV_2(x, 0, y_2, 0)\big)\Big)\Big)\Big)$$ — наречие определительное или одновременно обстоятельственное и определительное — наречие, принадлежащее одной из групп а)—в).

Можно переписать эту формулу в эквивалентном виде без импликации.

2. $AdV_3(x, y)$ — степени сравнения, где y = «сравн», если наречие в сравнительной степени, y = «прев», если наречие в превосходной степени.

Для сравнительной степени существуют две формы выражения $AdV_4(x, y)$, где

а) y = «синт», если форма выражения сравнительной степени синтетическая (простая), т.е. образуется с помощью суффиксов ***-ее (-ей), -ше*** и ***-е*** (*меньше, менее* и др.);

б) y = «анлт», если форма выражения сравнительной степени аналитическая, т.е. представляет собой сочетание слова *более* с исходной формой наречия (*более смело* и др.).

$(\forall x)\Big(AdV_3(x, сравн) \leftrightarrow \big(AdV_4(x, синт) \vee AdV_4(x, анлт)\big)\Big)$ — если наречие в сравнительной степени, то обязательно в одной из двух форм, и наоборот.

Превосходная степень наречий имеет две формы выражения $AdV_5(x, y)$, где

а) y = «синт», если форма выражения превосходной степени наречия синтетическая, т.е. образуется с помощью суффиксов ***-ейш(-е), -айш(-е)*** (***-е*** — суффикс наречия) (*покорнейше* и др.). Такая форма устарела и теперь употребляется крайне редко;

б) y = «анлт», если форма выражения превосходной степени наречия аналитическая, т.е. образуется

$AdV_5^1(x)$ — из сочетания формы сравнительной степени со словом *всех* или *всего* (*громче всех* и др.);

$AdV_5^2(x)$ — из сочетания слова *наиболее* с исходной формой наречия (*наиболее целесообразно* и др.).

$(\forall x)\Big(AdV_5(x, анлт) \leftrightarrow \Big(\big(AdV_5^1(x) \,\&\, \neg AdV_5^2(x)\big) \vee \big(AdV_5^2(x) \,\&\, \neg AdV_5^1(x)\big)\Big)\Big)$ — если б), то одна из последних двух, и наоборот.

$(\forall x)\big(AdV_3(x, прев) \leftrightarrow \big(AdV_5(x, синт) \vee AdV_5(x, анлт)\big)\big)$ — если наречие в превосходной степени, то обязательно в одной из двух форм, и наоборот.

$(\forall x)\big(\big(AdV_3(x, сравн) \vee AdV_3(x, прев)\big) \to \big(AdV_2(x, y_1, 0, 0) \vee AdV_2(x, y_1, y_2, 0)\big)\big)$ — степени сравнения существуют только для качественных наречий или наречий, являющихся одновременно качественными и количественными. Можно записать симметричную формулу, что для количественных и наречий способа и образа действия не существует степени сравнения.

3. $AdV_6(x)$ — степени качества (формы субъективной оценки), в отличие от степеней сравнения, выражают меру признака безотносительно к сравнению. К степеням качества относятся:

а) $AdV_6^1(x)$ — приставочные образования (*презабавно* и др.);

б) $AdV_6^2(x)$ — суффиксальные образования (*плоховато* и др.);

в) $AdV_6^3(x)$ — сочетания наречий меры и степени с исходной формой (*очень красиво* и др.);

г) $AdV_6^4(x)$ — удвоение наречий (*далеко-далеко* и др.).

Принадлежность наречия к одной из четырёх групп можно определить только по словообразованию, перечислив приставки и суффиксы, при помощи которых образуется степень качества.

$$(\forall x)\left(AdV_6(x) \leftrightarrow \underset{\substack{i=1\\ i\neq j}}{\overset{4}{\&}}\left(\neg\left(AdV_6^i(x) \to AdV_6^j(x)\right)\vee\neg\left(AdV_6^j(x) \to AdV_6^i(x)\right)\right)\right) —$$

а)—г) ↔ степень качества.

$(\forall x)\left(AdV_6(x) \to \left(AdV_2(x, y_1, 0, 0)\vee AdV_2(x, y_1, y_2, 0)\right)\right)$ — степени качества существуют только для качественных наречий или наречий, являющихся одновременно качественными и количественными. Можно записать симметричную формулу, что для количественных и наречий способа и образа действия не существует степени качества.

2.9. Предлог

1. $Prep_1(x, y)$ — по происхождению предлоги делятся на y = «непр», т.е. x — непроизводный (первообразный) предлог (*в, под, про* и др.) и y = «пр», т.е. x — производный предлог. Производные предлоги делятся на

а) $Prep_1^1(x)$ — отнаречные (*близ, около, сквозь* и др.);

б) $Prep_1^2(x)$ — отыменные (*вследствие, по пути, по причине* и др.);

в) $Prep_1^3(x)$ — отглагольные (*благодаря, несмотря на, спустя* и др.).

$$(\forall x)\left(Prep_1(x, np) \leftrightarrow \underset{\substack{i=1\\ i\neq j}}{\overset{3}{\&}}\left(\neg\left(Prep_1^i(x) \to Prep_1^j(x)\right)\vee\neg\left(Prep_1^j(x) \to Prep_1^i(x)\right)\right)\right) —$$

производный предлог ↔ он принадлежит одной из групп а)—в).

2. $Prep_2(x, y)$ — с точки зрения структуры выделяются предлоги: y = «прост», т.е. простые (*по, мимо* и др.); y = «сл», т.е. сложные (парные) (*из-за, из-под* и др.); y = «сост», т.е. составные (*в деле, в отношении* и др.); y = «слст», т.е. сложно-составные (*в отличие от, наравне с* и др.).

3. $Prep_3(x, y)$ — отношения, выражаемые предлогами: y = «прстр», т.е. пространственные (*из, к, вдоль, вне, поверх* и др.); y = «вр», т.е. временное (*через, по, во время* и др.); y = «прич», т.е. причинное (*в силу, ввиду, благодаря* и др.); y = «цел», т.е. целевое (*для, за, по* и др.); y = «объек», т.е. объективное (*про, относительно, по* и др.).

2.10. Союз

1. $Con_1(x, y)$ — морфологическое строение союзов: y = «непр», если x — непроизводный (первообразный) союз (*а, но, и* и др.); y = «пр», если x — производный союз. Производные союзы делятся на

а) $Con_1^1(x)$ — простые (*что, как* и др.);

б) $Con_1^2(x)$ — составные (*потому что, после того как* и др.);

$(\forall x)\left(Con_1(x, пр) \leftrightarrow \left(\left(Con_1^1(x) \,\&\, \neg Con_1^2(x)\right) \vee \left(Con_1^2(x) \,\&\, \neg Con_1^1(x)\right)\right)\right)$ — производный союз ↔ он принадлежит одной из групп: а) или б). Эта формула может быть записана в эквивалентном виде с импликацией в правой части.

2. $Con_2(x, y)$ — с точки зрения структуры выделяются союзы: y = «один», т.е. x — одиночный союз (*и, что, будто* и др.); y = «повт», т.е. x — повторяющийся союз (*и—и, ни—ни* и др.); y = «парн», т.е. x — двойной (парный) союз (*если—то, насколько—настолько* и др.).

3. $Con_3(x, y)$ — синтаксические функции союзов: y = «соч», если x — сочинительный союз, y = «пдч», если x — подчинительный союз.

Сочинительные союзы делятся на

а) $Con_3^1(x, y)$ — соединительные (*ни—ни, да* и др.);

б) $Con_3^2(x, y)$ — разделительные (*либо, то—то* и др.);

в) $Con_3^3(x, y)$ — противительные (*но, зато, однако* и др.).

$$(\forall x)\left(Con_3(x, соч) \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i \neq j}}^{3} \left(\neg\left(Con_3^i(x) \to Con_3^j(x)\right) \vee \neg\left(Con_3^j(x) \to Con_3^i(x)\right)\right)\right)$$ — сочинительные союзы — союзы, принадлежащее одной из групп а)—в).

Эквивалентная запись этой формулы:

$$(\forall x)\left(Con_3(x, соч) \leftrightarrow \mathop{\&}_{\substack{i=1 \\ i \neq j}}^{3} \left(\left(Con_3^i(x) \,\&\, \neg Con_3^j(x)\right) \vee \left(Con_3^j(x) \,\&\, \neg Con_3^i(x)\right)\right)\right).$$

Подчинительные союзы делятся на

а) $Con_3^4(x, y)$ — временные (*когда, пока* и др.);

б) $Con_3^5(x, y)$ — сравнительные (*как, будто, словно* и др.);

в) $Con_3^6(x,y)$ — целевые (*чтобы, дабы* и др.);

г) $Con_3^7(x,y)$ — уступительные (*хотя, несмотря на то что* и др.).

$$(\forall x)\left(Con_3(x, подч) \leftrightarrow \mathop{\&}_{\substack{i=4\\ i\neq j}}^{7}\left(\neg\left(Con_3^i(x)\to Con_3^j(x)\right)\vee\neg\left(Con_3^j(x)\to Con_3^i(x)\right)\right)\right) —$$

подчинительные союзы — союзы, принадлежащее одной из групп а)—г).

Эквивалентная запись этой формулы:

$$(\forall x)\left(Con_3(x, подч) \leftrightarrow \mathop{\&}_{\substack{i=4\\ i\neq j}}^{7}\left(\left(Con_3^i(x)\,\&\,\neg Con_3^j(x)\right)\vee\left(Con_3^j(x)\,\&\,\neg Con_3^i(x)\right)\right)\right).$$

2.11. Частица

1. $PartL_1(x,y)$ — разряды частиц по значению: y = «смысл», если частица выражает смысловые оттенки значений; y = «эмоц», если частица вносит эмоционально-экспрессивный оттенок (*ведь, ну и, то—то* и др.); y = «мод», если x — модальная частица; y = «слобр», если x — словообразующая частица (*-то, -ка* и др.) или формообразующая частица (*бы (б), да, пусть, пускай* и др.).

Частицы, выражающие смысловые оттенки значений делятся на

а) $PartL_1^1(x)$ — указательные (*вот, это, во* и др.);

б) $PartL_1^2(x)$ — определительные, т.е. служат для уточнения смысла (*именно, как раз, почти, просто* и др.);

в) $PartL_1^3(x)$ — выделительно-ограничительные (*всё, исключительно, разве лишь, только* и др.);

г) $PartL_1^4(x)$ — усилительные (*уже, ещё, даже* и др.).

$$(\forall x)\left(PartL_1(x, смысл) \leftrightarrow \mathop{\&}_{\substack{i=1\\ i\neq j}}^{4}\left(\neg\left(PartL_1^i(x)\to PartL_1^j(x)\right)\vee\neg\left(PartL_1^j(x)\to PartL_1^i(x)\right)\right)\right) —$$

смысловые частицы — частицы, принадлежащие одной из групп а)—г).

Эквивалентная запись этой формулы:

$$(\forall x)\left(PartL_1(x, смысл) \leftrightarrow \underset{\substack{i=1 \\ i\neq j}}{\overset{4}{\&}}\left(\left(PartL_1^i(x) \,\&\, \neg PartL_1^j(x)\right) \vee \left(PartL_1^j(x) \,\&\, \neg PartL_1^i(x)\right)\right)\right).$$

Модальные частицы делятся на

а) $PartL_1^5(x)$ — модально-волевые (*ну, дай, пусть* и др.);

б) $PartL_1^6(x)$ — утвердительные (*да, точно* и др.);

в) $PartL_1^7(x)$ — вопросительные (*ли, а, неужели* и др.);

г) $PartL_1^8(x)$ — отрицательные (*не, ни, нет*);

д) $PartL_1^9(x)$ — частицы, служащие для передачи и оценки чужой речи (*мол, дескать* и др.).

$$(\forall x)\left(PartL_1(x, мод) \leftrightarrow \underset{\substack{i=5 \\ i\neq j}}{\overset{9}{\&}}\left(\neg\left(PartL_1^i(x) \to PartL_1^j(x)\right) \vee \neg\left(PartL_1^j(x) \to PartL_1^i(x)\right)\right)\right)$$ —

модальные частицы — частицы, принадлежащее одной из групп а)—д).

Эквивалентная запись этой формулы:

$$(\forall x)\left(PartL_1(x, мод) \leftrightarrow \underset{\substack{i=5 \\ i\neq j}}{\overset{9}{\&}}\left(\left(PartL_1^i(x) \,\&\, \neg PartL_1^j(x)\right) \vee \left(PartL_1^j(x) \,\&\, \neg PartL_1^i(x)\right)\right)\right)$$

.

Ясно, что формула $\underset{i\neq j}{\&}\left(\left(P_i \,\&\, \neg P_j\right) \vee \left(P_j \,\&\, \neg P_i\right)\right)$ эквивалентна каждой из следующих формул:

$$\underset{i\neq j}{\&}\left(\neg\left(P_i \to P_j\right) \vee \neg\left(P_j \to P_i\right)\right),$$

$$\underset{i\neq j}{\&}\neg\left(\left(P_i \to P_j\right) \& \left(P_j \to P_i\right)\right),$$

$$\neg \underset{i\neq j}{\vee}\left(\left(P_i \to P_j\right) \& \left(P_j \to P_i\right)\right).$$

Это можно доказать, применяя правила:

$$A \to B \equiv \neg A \vee B\,,$$
$$\neg\left(A \,\&\, B\right) \equiv \neg A \vee \neg B\,,$$
$$\neg\left(A \vee B\right) \equiv \neg A \,\&\, \neg B\,.$$

Чтобы некоторые формулы записывались короче, можно ввести следующее обозначение. Пусть некоторая часть речи

$$S \in \left\{N, Adj, Num, ProN, V, PartP, VA, AdV, Prep, Con, PartL\right\},$$

т.е. совокупность слов; причём это множество слов по определённому признаку разбивается на непересекающиеся множества, на которых истинны предикаты $M_1,...,M_l$. И пусть φ — произвольная формула узкого исчисления предикатов. Пусть истинна формула

$$\left(M_2 \to \varphi\right) \& \left(\left(M_1 \vee M_3 \vee M_4 \vee ... \vee M_l\right) \to \neg\varphi\right).$$

Тогда в общем случае определим

$$\Phi\left[i, S, \varphi\right] \overset{df}{=} \left(\left(M_i \to \varphi\right) \& \left(\bigvee_{j=1,\, j\neq i}^{l} M_j \to \neg\varphi\right)\right).$$

Для формулы, приведенной выше, имеем $i = 2$. В частном случае (при $i = 1$) для краткости будем писать $\Phi\left[S, \varphi\right]$.

Например, если положим

$$M_1 = Adj_1(x, кач)\,,\ M_2 = Adj_1(x, отнс)\,,\ M_3 = Adj_1(x, прит)\,,\ S = Adj$$

и возьмём $\varphi = \left(Adj_{10}(x, полн) \vee Adj_{10}(x, кр)\right)$, то краткая запись будет $\Phi\left[S, \varphi\right]$.

3. СИНТАКСИЧЕСКИЕ ПРЕДИКАТЫ

Определим одноместные предикаты членов предложения: $P_{sub}(x)$, где x — подлежащее; $P_{pred}(x)$, где x — сказуемое; $P_{attr}(x)$, где x — определение; $P_{obj}(x)$, где x — дополнение; $P_{adv}(x)$, где x — обстоятельство.

Ещё введём двуместные предикаты членов предложения: $P_{sub}(x,y)$, x — подлежащее; $P_{pred}(x,y)$, x — сказуемое; $P_{attr}(x,y)$, x — определение; $P_{obj}(x,y)$, x — дополнение; $P_{adv}(x,y)$, x — обстоятельство; где y играет роль определяемого (поясняемого) слова или словосочетания (т.е. слова или словосочетания, от которого задаётся вопрос к тому или иному члену предложения).

Для обозначения однородных членов предложения (т.е. тех членов предложения, которые относятся к одному слову и отвечают на один и тот же вопрос) введём предикаты $P_{homo}(x_1,...,x_n)$, где $x_1,...,x_n$ — однородные члены и $P_{homo}(x_1,...,x_n,y)$, при этом y — слово или словосочетание, к которому относятся $x_1,...,x_n$.

Пусть у нас есть предикат $Sent_{com}(x_1,...,x_n)$ для определения сложного предложения с союзами $x_1,...,x_n$. Условимся, что составные части этого сложного предложения (главные и придаточные) начинаются с союза. Если предложение бессоюзное, то вместо x_i пишем «0».

Если истинен предикат $Con_2(x,повт)$ или $Con_2(x,парн)$, т.е. $x = x_1 + x_2$ — повторяющийся или парный союз соответственно (например, «не только … но и»), составные части которого x_1 и x_2, то будем считать, что определены предикаты $Con_2(x_1,повт)$ и $Con_2(x_2,повт)$ или $Con_2(x_1,парн)$ и $Con_2(x_2,парн)$. Так как с точки зрения синтаксиса составные части таких союзов не являются самостоятельными союзами, то подразумеваем, что последние два предиката введены чисто условно, для удобства в реализации программы.

На логическом уровне мы предполагаем, что выполняются

$$\forall x\ P_{sub}(x) \leftrightarrow \neg P_{pred}(x)$$
$$\forall x\ P_{sub}(x) \leftrightarrow \neg P_{obj}(x)$$
$$\forall x\ P_{sub}(x) \leftrightarrow \neg P_{attr}(x)$$
$$\forall x\ P_{sub}(x) \leftrightarrow \neg P_{adv}(x)$$

$$\forall x, y\ P_{sub}(x,y) \leftrightarrow P_{pred}(y,x) \qquad (*)$$
$$\forall x, y\ P_{sub}(x,y) \leftrightarrow \neg P_{pred}(x,y)$$
$$\forall x, y\ P_{sub}(x,y) \leftrightarrow \neg P_{obj}(x,y)$$
$$\forall x, y\ P_{sub}(x,y) \leftrightarrow \neg P_{attr}(x,y) \quad \text{и т.д.}$$
$$\forall x, y\ P_{sub}(x,y) \leftrightarrow \neg P_{adv}(x,y)$$

Но для того, чтобы запрограммировать предикативное представление, мы не будем хранить эти данные как дополнительную информацию, т.е. будем считать, что эти требования выполняются по умолчанию.

Теперь можно записать формульное представление свойств этих предикатов, считая, что *x, y* — слова или словосочетания. Верхний индекс в скобках при Q — местность предиката, нижний индекс Q является показателем, от какого члена предложения задается вопрос.

1. Определяемое слово является подлежащим

а) $\left(\forall x, y\right)\left(Q_1^{(2)}(x,y) \leftrightarrow \left(P_{sub}(x,y)\,\&\,P_{sub}(x)\,\&\,P_{pred}(y)\right)\right)$ — от подлежащего можно задать вопрос к сказуемому;

а') если к этой формуле применить (*), т.е. заменить $P_{sub}(x,y)$ на $P_{pred}(y,x)$, то формула останется верной:

$$\left(\forall x, y\right)\left(Q_1^{(2)}(x,y) \leftrightarrow \left(P_{pred}(y,x)\,\&\,P_{sub}(x)\,\&\,P_{pred}(y)\right)\right);$$

б) $\left(\forall x, y\right)\left(Q_1^{(2)}(x,y) \leftrightarrow \left(P_{attr}(y,x)\,\&\,P_{sub}(x)\,\&\,P_{attr}(y)\right)\right)$ — от подлежащего можно задать вопрос к определению;

в) $\left(\forall x, y\right)\left(Q_1^{(2)}(x,y) \leftrightarrow \left(P_{obj}(y,x)\,\&\,P_{sub}(x)\,\&\,P_{obj}(y)\right)\right)$ — от подлежащего можно задать вопрос к дополнению.

2. Определяемое слово является сказуемым

а) $\left(\forall x, y\right)\left(Q_2^{(2)}(x,y) \leftrightarrow \left(P_{pred}(x,y)\,\&\,P_{pred}(x)\,\&\,P_{sub}(y)\right)\right)$ — от сказуемого можно задать вопрос к подлежащему;

а') $\left(\forall x, y\right)\left(Q_2^{(2)}(x,y) \leftrightarrow \left(P_{sub}(y,x)\,\&\,P_{pred}(x)\,\&\,P_{sub}(y)\right)\right)$;

б) $\left(\forall x, y\right)\left(Q_2^{(2)}(x,y) \leftrightarrow \left(P_{obj}(y,x)\,\&\,P_{pred}(x)\,\&\,P_{obj}(y)\right)\right)$ — от сказуемого можно задать вопрос к дополнению;

в) $\left(\forall x, y\right)\left(Q_2^{(2)}(x,y) \leftrightarrow \left(P_{adv}(y,x) \,\&\, P_{pred}(x) \,\&\, P_{adv}(y)\right)\right)$ — от сказуемого можно задать вопрос к обстоятельству;

г) $\left(\forall x, y\right)\left(Q_2^{(2)}(x,y) \leftrightarrow \left(P_{attr}(y,x) \,\&\, P_{pred}(x) \,\&\, P_{attr}(y)\right)\right)$ — от сказуемого можно задать вопрос к определению.

3. Определяемое слово является дополнением

а) $\left(\forall x, y\right)\left(Q_3^{(2)}(x,y) \leftrightarrow \left(P_{attr}(y,x) \,\&\, P_{obj}(x) \,\&\, P_{attr}(y)\right)\right)$ — от дополнения можно задать вопрос к определению;

б) $\left(\forall x, y\right)\left(Q_3^{(2)}(x,y) \leftrightarrow \left(P_{obj}(y,x) \,\&\, P_{obj}(x) \,\&\, P_{obj}(y)\right)\right)$ — от дополнения можно задать вопрос к дополнению.

4. Определяемое слово является обстоятельством

а) $\left(\forall x, y\right)\left(Q_4^{(2)}(x,y) \leftrightarrow \left(P_{obj}(y,x) \,\&\, P_{adv}(x) \,\&\, P_{obj}(y)\right)\right)$ — от обстоятельства можно задать вопрос к дополнению;

б) $\left(\forall x, y\right)\left(Q_4^{(2)}(x,y) \leftrightarrow \left(P_{attr}(y,x) \,\&\, P_{adv}(x) \,\&\, P_{attr}(y)\right)\right)$ — от обстоятельства можно задать вопрос к определению.

5. Определяемое слово является определением

а) $\left(\forall x, y\right)\left(Q_5^{(2)}(x,y) \leftrightarrow \left(P_{obj}(y,x) \,\&\, P_{attr}(x) \,\&\, P_{obj}(y)\right)\right)$ — от определения можно задать вопрос к дополнению.

Помимо двуместных предикатов можно ввести многоместные. Это возможно, если в предложении от одного члена предложения можно задать вопросы к нескольким одинаковым членам предложения, причём последние не должны являться однородными (т.е. должны отвечать на разные вопросы, либо характеризовать предмет или действие с разных сторон). Так как из основ синтаксиса известно, что в простом предложении не может быть несколько неоднородных подлежащих и сказуемых, то для реализации этого случая остаются предложения с неоднородными дополнениями, определениями и обстоятельствами. Например, для следующего предложения истинна формула с трёхместным предикатом: *Купить машину нам не по средствам.*

$$\left(\forall x, y_1, y_2\right)$$

$$\left(Q_2^{(3)}(x, y_1, y_2) \leftrightarrow \left(P_{obj}(y_1, x) \,\&\, P_{obj}(y_2, x) \,\&\, P_{pred}(x) \,\&\, P_{obj}(y_1) \,\&\, P_{obj}(y_2)\right)\right),$$

если положить x = «купить», y_1 = «машину», y_2 = «нам».

В общем виде формулы для n неоднородных членов предложения записываются следующим образом:

$$\left(\forall x, y_1, ..., y_n\right)\left(Q_1^{(n+1)}(x, y_1, ..., y_n) \leftrightarrow \left(\mathop{\&}_{i=1}^{n} P_{attr}(y_i, x) \,\&\, P_{sub}(x) \,\&\, \mathop{\&}_{i=1}^{n} P_{attr}(y_i)\right)\right) \quad —$$

неоднородные определения при подлежащем.

$$\left(\forall x, y_1, ..., y_n\right)\left(Q_1^{(n+1)}(x, y_1, ..., y_n) \leftrightarrow \left(\mathop{\&}_{i=1}^{n} P_{obj}(y_i, x) \,\&\, P_{sub}(x) \,\&\, \mathop{\&}_{i=1}^{n} P_{obj}(y_i)\right)\right) \quad —$$

неоднородные дополнения при подлежащем.

$$\left(\forall x, y_1, ..., y_n\right)\left(Q_2^{(n+1)}(x, y_1, ..., y_n) \leftrightarrow \left(\mathop{\&}_{i=1}^{n} P_{obj}(y_i, x) \,\&\, P_{pred}(x) \,\&\, \mathop{\&}_{i=1}^{n} P_{obj}(y_i)\right)\right) \quad —$$

неоднородные дополнения при сказуемом.

$$\left(\forall x, y_1, ..., y_n\right)\left(Q_2^{(n+1)}(x, y_1, ..., y_n) \leftrightarrow \left(\mathop{\&}_{i=1}^{n} P_{adv}(y_i, x) \,\&\, P_{pred}(x) \,\&\, \mathop{\&}_{i=1}^{n} P_{adv}(y_i)\right)\right) —$$

неоднородные обстоятельства при сказуемом.

$$\left(\forall x, y_1, ..., y_n\right)\left(Q_3^{(n+1)}(x, y_1, ..., y_n) \leftrightarrow \left(\mathop{\&}_{i=1}^{n} P_{attr}(y_i, x) \,\&\, P_{obj}(x) \,\&\, \mathop{\&}_{i=1}^{n} P_{attr}(y_i)\right)\right) \quad —$$

неоднородные определения при дополнении.

$$\left(\forall x, y_1, ..., y_n\right)\left(Q_4^{(n+1)}(x, y_1, ..., y_n) \leftrightarrow \left(\mathop{\&}_{i=1}^{n} P_{attr}(y_i, x) \,\&\, P_{adv}(x) \,\&\, \mathop{\&}_{i=1}^{n} P_{attr}(y_i)\right)\right) \quad —$$

неоднородные определения при обстоятельстве.

Проиллюстрируем на примерах, как определяются предикаты.

Например, для предложения типа «Существительное (в ИП) + Глагол (его спрягаемая форма)»: *Грачи прилетели.*

Представление в виде предикатов будет: $P_{sub}(грачи)$, $P_{pred}(прилетели)$, $P_{sub}(грачи, прилетели)$, $P_{pred}(прилетели, грачи)$; и есть формульное представление предикатов 1. а), а') — если положить x = «грачи», y = «прилетели»; 2. а), а') — если положить x = «прилетели», y = «грачи».

Или еще один пример предложения типа «Глагол (инфинитив) + Глагол-связка «быть» (его спрягаемая форма) + Существительное (в беспредложной или предложной форме любого косвенного падежа, которая способна сочетаться с глаголом-связкой «быть») или Наречие (способное сочетаться с глаголом-связкой «быть»)»: *Купить машину нам не по средствам.*

Представление в виде предикатов:

$P_{pred}(купить)$, $P_{obj}(машину)$, $P_{obj}(нам)$, $P_{adv}(не\ по\ средствам)$,

$P_{obj}(машину, купить)$, $P_{obj}(нам, купить)$, $P_{adv}(не\ по\ средствам, купить)$;

формульное представление предикатов 2. б) — если положить x = «купить», y = «машину» или y = «нам», 2. в) — если положить x = «купить», y = «не по средствам».

Для написания программы и просто для наглядности информацию о взаимосвязи между предикатами удобно представить в виде таблиц.

Таблица 1

«От 1 можно задать вопрос к 2»[1]

1	2
подлежащее	сказуемое
	определение
	дополнение
сказуемое	подлежащее
	дополнение
	обстоятельство
	определение
дополнение	определение
	дополнение
обстоятельство	дополнение
	определение
определение	дополнение

[1] Под словами «можно задать вопрос» подразумевается, что между словами 1 и 2 есть односторонняя связь. (Эта таблица является более наглядным представлением формул, введённых на с. 2—3.)

Таблица 2

«1 может быть выражено посредством 2»

1	2
подлежащее	существительное в ИП
	глагол (инфинитив)
	прилагательное
	причастие
	числительное
	наречие
	союзы, частицы (если они в кавычках)
	словосочетание: а) сущ./мест. в ИП + сущ./мест. в ТП б) числ./сущ. в ИП + сущ в РП в) числ./прил. + «из» + сущ./прил./числ. в РП
сказуемое	глагол (в любой форме)
	вспомогательный глагол[2] + глагол (инфинитив)
	а) связка[3]/– + сущ. в РП без предлога, ИП, ТП, в ост. падежах с предлогом и без б) связка/– + прил. в краткой форме в) связка/– + прил. в полной форме в ИП, ТП г) связка/– + прил. в сравнительной и превосходной степени д) связка/– + прил. в полной форме + зависимые слова (т.е. слова, к которым можно задать вопрос от этого прил.) е) прил. в краткой форме + глагол (инфинитив) ж) связка/– + причастие в полной и краткой форме з) связка/– + наречие и) связка/– + числительное к) связка/– + числ. в ИП + сущ. в РП

[2] — [6] с. 419.
[3] — [6] с. 421.

определение	прилагательное
	причастие
	причастный оборот
	сущ./ местоим. в РП и ТП без предлога, в остальных падежах (кроме ИП) с предлогом
	глагол (инфинитив)
	неделимое сочетание слов: прил. + сущ.
обстоятельство	наречие
	деепричастие
	деепричастный оборот
	глагол (инфинитив, который относится к личной форме глагола/прич./деепр.)
	словосочетания: а) сущ./ мест. + сущ. б) наречие + сущ.
	сущ. в любом падеже (кроме ИП) с предлогом и без, которое относится к личной форме глагола/прич./деепр.; часто в сочетании с согласованным в роде, числе и ТП определением без предлога; в сочетании с *несмотря на, навзирая на, в случае*
дополнение	сущ./мест. в РП и ВП без предлога — прямое дополнение; сущ./мест. в ВП с предлогами *в, за, на, о, под, про*; РП, ДП и ТП без предлогов; РП с предлогами *из, от, с, у, для, до, без*, ДП с предлогом *к*, ТП с предлогами *за, с, над, перед*, ПП с предлогами *о, в, на* — косвенное дополнение;
	числ. в ИП + сущ. в РП
	причастие
	глагол (инфинитив)
	сущ./мест. + сущ./ мест./ числ.

Таблица 3

«Часть речи 1 может являться членом предложения 2»

1	2
существительное	подлежащее
	сказуемое
	дополнение
	обстоятельство
	определение
прилагательное	подлежащее
	сказуемое
	определение
числительное	подлежащее
	сказуемое
местоимение	подлежащее
	дополнение
	определение
глагол	подлежащее (инфинитив)
	сказуемое
	дополнение (инфинитив)
	обстоятельство (инфинитив)
	определение (инфинитив)
причастие	подлежащее
	сказуемое
	дополнение
	определение
деепричастие	обстоятельство
наречие	подлежащее (если в «»)
	сказуемое
	обстоятельство

4. ВАЛЕНТНОСТИ СЛОВА

4.1. Семантические валентности

Валентностями обладают слова, которые являются предикатами, т. е. те, которые задают ситуацию — это все глаголы, некоторые существительные

(отглагольные), прилагательные (обозначающие сравнение: больше, меньше, выше, ниже), некоторые предлоги и наречия.

Валентности слова бывают синтаксические и семантические. Семантические валентности определяются лексическим анализом ситуации, задаваемой этим словом. Приведём пример со словом *аренда* или *арендовать*. A *арендует* C значит, в первом приближении, что за какое-то вознаграждение D лицо A приобретает у другого лица B право на эксплуатацию недвижимой собственности C в течении времени T. Следовательно, существенными для ситуации аренды являются следующие «участники» или семантические актанты: субъект аренды (тот, кто арендует), первый объект аренды (то, что арендуют), контрагент (тот, у кого арендуют), второй объект (плата) и срок.

Эти актанты необходимы, так как устранение какого-либо из них изменяет смысл ситуации. Например, если убрать срок, то ситуация аренды трансформируется в ситуацию купли-продажи. С другой стороны, эти актанты достаточны, поскольку в ситуации аренды не требуется указание того, по какой причине, где, когда и с какой целью она осуществлялась. Хотя соответствующие словоформы грамматически присоединимы к глаголу *арендовать* [1].

В итоге, эта ситуация имеет 5 валентностей и формально записывается в виде предиката $P^{val}(y, x_1, x_2, x_3, x_4, x_5)$, где x_1 — «кто», x_2 — «что», x_3 — «у кого», x_4 — «цена», x_5 — «срок».

В предложении могут быть определены актанты не для всех семантических валентностей, некоторые могут просто не упоминаться или вообще не иметь синтаксического выражения.

4.2. Синтаксические валентности

Синтаксические валентности — это те, которые представлены в тексте. Они определяются присоединяемыми к слову подлежащими и дополнениями и зависят от контекста.

Например, глагол *промахнуться* имеет 4 семантические валентности: кто (деятель), во что/по чему (мишень), из чего (оружие — факультативно) и чем (орган, средство). Но в большинстве контекстов синтаксически выражается лишь первая валентность. Например, нельзя сказать *«Он промахнулся в окно бутылкой»*.

Возможны случаи, когда синтаксических валентностей у слова больше, чем семантических.

Для того чтобы не связывать с каждым глаголом (и другими словами) отдельный предикат, будем рассматривать предикат, размерность которого больше на 1: $P^{val}(y, x_1, x_2, \ldots, x_n)$, при этом y будет само слово, а $x_1, x_2, \ldots, x_n$ — его валентности. Чтобы отличать синтаксические и семантические актанты можно использовать мультииндексы, чтобы указать, какие актанты заданы в тексте. Запись $P^{val}_{i_1 i_2 \ldots i_k}(y, x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ означает, что заданы актанты $i_1, i_2, \ldots, i_k$. В частности, если заданы все актанты, то получаем $P^{val}_{1\ldots n}(y, x_1, x_2, \ldots, x_n)$. Некоторые варианты (наборов мультииндексов) могут быть недопустимы в языке. Если набор $i_1, i_2, \ldots, i_k$ допустим, то имеет место импликация:

$$\forall y \forall x_1 \ldots \forall x_n (P^{val}_{1\ldots n}(y, x_1, x_2, \ldots, x_n) \to P^{val}_{i_1 i_2 \ldots i_k}(y, x_{i_1}, x_{i_2}, \ldots, x_{i_k})).$$

Более того, если имеется два набора допустимых мультииндексов $\langle i_1, i_2, \ldots, i_k \rangle$ и $\left\langle i_1', i_2', \ldots, i_s' \right\rangle$ таких, что $\langle i_1, i_2, \ldots, i_k \rangle \supseteq \left\langle i_1', i_2', \ldots, i_s' \right\rangle$, то имеет место аналогичная импликация:

$$\forall y \forall x_{i_1} \ldots \forall x_{i_k} \forall x_{i_1'} \ldots \forall x_{i_s'} (P^{val}_{i_1 \ldots i_k}(y, x_{i_1}, x_{i_2}, \ldots, x_{i_k}) \to P^{val}_{i_1' i_2' \ldots i_s'}(y, x_{i_1'}, x_{i_2'}, \ldots, x_{i_s'})).$$

Имеются следующие факты из синтаксиса [6].

Составное глагольное сказуемое (СГС) состоит из вспомогательного глагола и инфинитива, т.е. содержательной (главной) части. Для составного глагольного сказуемого вспомогательными могут выступать глаголы со стр.15.

Составное именное сказуемое (СИС) состоит из связки и именной части, выраженной именем существительным, именем прилагательным, причастием, именем числительным, местоимением, наречием или междометием. Для составного именного сказуемого глаголы-связки могут быть глаголами (стр. 15–16).

Сложным называется такое сказуемое, которое состоит из трёх или более слов и соединяет в себе, как правило, признаки СГС и СИС.

При определении синтаксической валентности глагола в предложении мы имеем дело с тремя ситуациями.

Первая из них — когда в предложении глагол входит в состав простого глагольного или в СИС. Если сказуемые однородные, то определяем валентность только одного из них, валентности других будут такими же. Ва-

лентность и соответствующие актанты такого глагола легко определить по количеству вопросов, задаваемых от него.

Вторая ситуация — глагол входит в сложное сказуемое или в СГС. Часть (1) связана с подлежащим, а (2) имеет связь с остальными членами предложения. Иными словами, вспомогательный глагол имеет валентность, равную 1 (если предложение не безличное), а содержательная часть СГС имеет валентность, равную числу вопросов, задаваемых от сказуемого. В терминах синтаксических предикатов, которые были введены ранее, это означает, что валентность главной части СГС равна числу определённых в данном предложении предикатов, на втором месте у которых сказуемое, а на первом не подлежащее. В сложном сказуемом валентность вспомогательного глагола определяется, как в СГС, а число второстепенных членов предложения, связанных со сложным сказуемым, — есть валентность последнего из глаголов содержательной части.

Третья ситуация — глагол является второстепенным членом предложения. Здесь для определения валентности и актантов поступаем, как в первом случае. Просто в третьей ситуации валентность глагола очень редко будет превышать 1 (если вообще когда-нибудь будет превышать).

Значит, теперь мы можем составить таблицу актантов для каждого глагола в предложении и знаем валентности этих глаголов.

5. СТРУКТУРЫ, СООТВЕТСТВУЮЩИЕ ПРЕДЛОЖЕНИЯМ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Предложению на естественном языке сопоставим набор структур, состоящих из кортежей, которые в конечном итоге определят набор предикатов.

Первоначально предикаты рассматриваем на синтаксическом уровне, т.е. как записи. В дельнейшем на основе полученных структур будут конструироваться модели, т.е. будет осуществлён переход на семантический уровень. Под предикатами в этом случае будем понимать подмножества в соответствующих декартовых степенях основных множеств моделей.

Так как вводные слова и вводные конструкции требуют отдельного, более глубокого анализа, то будем их исключать при построении структур. Т.е. будем считать, что в предложении нет вводных слов или вводных структур. Это нужно для того, чтобы не возникло путаницы в случае, когда вводное слово можно перепутать с какой-то частью речи. Например: *Гово-*

рят, тут жить можно («говорят» — вводное слово). *Они говорят* («говорят» — глагол).

Сначала рассмотрим структуры, которые могут быть в простых предложениях.

1. Структуры, соответствующие глаголам.

1.1. Во-первых, рассмотрим наиболее простой случай. Будем считать, что в предложении встречается только один глагол. Допустим также, что присутствуют несколько существительных в различных падежах, но в каждом падеже имеется не более одного существительного. Для простоты положим, что предлоги отсутствуют. Такому предложению может быть сопоставлена структура, показанная ниже.

V	NounNom	NounGen	NounDat	NounAcc	NounInstr	NounPrep

где

V — глагол;

NounNom (от англ. nominative) — существительное в именительном падеже, если имеется;

NounGen (от англ. genitive) — существительное в родительном падеже, если имеется;

NounDat (от англ. dative) — существительное в дательном падеже, если имеется;

NounAcc (от англ. accusative) — существительное в винительном падеже, если имеется;

NounInstr (от англ. instrumental) — существительное в творительном падеже, если имеется;

NounPrep (от англ. prepositional) — существительное в предложном падеже, если имеется.

В случае, когда существительное в данном падеже в предложении отсутствует, соответствующая позиция структуры может быть заполнена некоторой вспомогательной информацией.

Например, можно ввести специальные константы:

Some — существительное в данном падеже отсутствует, но в принципе оно может присутствовать. Т.е. если говорить в терминах валентностей, то это означает, что глагол имеет высокую валентность, но данный актант не заполнен в данном предложении;

Empty — существительное в данном падеже отсутствует, и ничего в принципе не может быть дополнено, иначе говоря, глагол не согласуется с существительным в данном падеже;

Unknown — существительное в данном падеже отсутствует, и не известно (точнее, мы не знаем в данный момент), можно ли согласовать соответствующее существительное.

Структуре, описанной выше, естественным образом соответствует предикат вида $P(v, n_1, ..., n_6)$. Где v — имя глагола; $n_1, ..., n_6$ — существительные. Предикат шестиместный, так как в русском языке имеется шесть падежей.

1.2. Теперь предположим, что в предложении могут быть несколько существительных в одном и том же падеже. Простейшая структура получается, если в позиции, соответствующей падежу, размещать список существительных.

В этом случае можно считать, что предикат, который будет сопоставляться структуре, имеет вид $P(v, n_{11}, ..., n_{1k}; n_{21}, ..., n_{2k}; ...; n_{61}, ..., n_{6k})$, где k — фиксированное число.

Сначала перечисляем все существительные в именительном падеже, потом — в родительном и т.д. Здесь k — верхняя граница количества существительных в одной падежной форме. По-видимому, достаточно положить $k = 4$.

1.3. Допустим, что в предложении есть предлоги. В первую очередь надо определить для каждого предлога, к какому существительному он относится.

Наиболее простой случай, когда предлог стоит непосредственно рядом с существительным или разделён с ним одним или несколькими прилагательными. Тогда в структуре просто добавляются предлоги.

Если в итоге предлоги и существительные соотнесены друг с другом, то возникает предикат типа

$$P(v, prep_{11}, n_{11}, ..., prep_{1k}, n_{1k}; ...; prep_{61}, n_{61}, ..., prep_{6k}, n_{6k}),$$

где $prep_{ij}$ – предлоги. Если предлог отсутствует, то этот факт фиксируем с помощью некоторой константы, например, Nil. В частности, при существительном в именительном падеже предлог отсутствует.

Более сложная ситуация возникает, если предлог отдалён от существительного посредством «распространённого определения». В этом случае рассматривается вопрос о согласовании предлога с существительными в том или ином падеже. Если же всё-таки не удаётся установить связь предлога с существительным, то можно прибегнуть к частотной совместимости слов.

1.4. Есть некоторые глаголы, которые сочетаются с существительными только с определённым предлогом. Для таких застывших форм «глагол + предлог» структура приобретает вид

V + Prep	NounNom	NounGen	NounDat	NounAcc	NounInstr	NounPrep

Здесь существительные в любом падеже стоят уже без предлогов. Иными словами, появится предикат $P_{prep}(v, n_{11},...,n_{1k};...;n_{61},...,n_{6k})$. Ясно, что $n_{11},...,n_{1k}$ — существительные в именительном падеже — отсутствуют.

1.5. Пусть в предложении встречаются два или более глаголов, идущие последовательно друг за другом. Тогда можно выделить несколько случаев.

А. Если глаголы записаны через запятую или два из них соединены союзом «и», причём они совпадают по форме (стоят в одинаковой спрягаемой форме или являются инфинитивами), времени, числу и лицу (если это можно определить), то эти глаголы являются однородными членами предложения, т.е. предложение рассматривается простое. Поэтому достаточно рассмотреть структуру ранее указанного вида, соответствующую одному из таких глаголов, а для остальных глаголов будет то же самое.

В этом случае, кроме того, может возникнуть ситуация, когда после запятых есть повторяющиеся союзы (*и*…, *и*…; *или*…, *или*… и т.д.). Тогда глаголы, являясь однородными членами предложения, вновь будут сопоставлены одинаковым структурам.

Б. Будем пока считать, что в простом предложении последовательно встречаются только два глагола. Пусть Inf — инфинитив глагола. V, как и ранее, обозначает вообще наличие глагола, т.е. при употреблении этого обозначения подчёркивается, что нам не существенно, в какой форме стоит глагол: в личной или в форме инфинитива. Теперь все возможные ситуации сочетаемости глагола с глаголом дают нам новые структуры вида

V	Inf

VRef	Inf

,

где VRef означает, что глагол является возвратным, т.е. для этого глагола в предикате возвратности переменная *ref* принимает значение, равное нулю. Предикаты, соответствующие этим структурам, имеют вид $P(VInf, Inf)$ и $P(VInfRef, Inf)$.

В. Если в предложении подряд идут несколько глаголов, то структуры в предыдущем пункте можно продолжить за счёт добавления нужного числа инфинитивов.

1.6. Если в предложении есть глагол и наречие (обозначим его AdV — от англ. adverb), относящееся к этому глаголу, то структура будет иметь вид:

V	AdV

.

Тогда имеем предикат $P(v, adv)$.

Примечание. Так как правила склонения для существительных и местоимений-существительных (*я, они, себя* и т.д.) одинаковые, то во всех структурах существительные могут быть заменены местоимениями-существительными.

2. Структуры, соответствующие прилагательным.

2.1. Пусть сначала в предложении есть полные прилагательные, и нет прилагательных в краткой форме, в сравнительной или превосходной степенях. Рассмотрим самый простой случай, когда в предложении одно существительное и прилагательное, которое с ним согласовано, и стоят они рядом. К тому же, так как известно, что род, число, падеж полного прилагательного определяются родом, числом и падежом относящегося к нему существительного, то структуру таких предложений можно представить в виде

Adj	N

,

где Adj (от англ. adjective) — прилагательное, N — существительное; род, число, падеж существительного и прилагательного совпадают. Поэтому получаем предикат вида $P(adj, n)$.

2.2. Для краткой формы прилагательного определить, к какому существительному оно относится, можно по роду и числу. Структура и соответствующий ей предикат при этом остаются без изменений.

2.3. Структура, соответствующая простой форме сравнительной степени прилагательного, имеет следующий вид

Compar	N	NounGen

,

где Compar (от англ. comparative) — прилагательное в простой форме сравнительной степени; N (от англ. noun) — существительное в любом падеже.

Этой структуре соответствует предикат $P(compar, n_1, n_2)$.

В случае, когда существительное в данном падеже отсутствует, соответствующая позиция структуры может быть заполнена специальной константой Nothing.

2.4. Идущие подряд прилагательные (в полной, краткой форме или в форме сравнительной степени) обозначают признак одного предмета, поэтому относятся к одному и тому же существительному и совпадают с ним по роду, числу и падежу (если соответствующие морфологические признаки имеют место).

В частности, к этому случаю можно отнести структуру, соответствующую сложной форме превосходной степени прилагательного п.а), а точнее её часть без существительного в родительном падеже с предлогом. В этой структуре «самый» и AdjC — два идущих подряд прилагательных, которые относятся к одному существительному Noun.

2.5. В предложениях естественного языка может встречаться согласование прилагательного и наречия. Поэтому необходимо рассматривать структуру вида

Adj	AdV

и соответствующий ей предикат $P(adj, adv)$.

Примечание: Во всех структурах вместо существительных могут стоять местоимения-существительные, а вместо прилагательных — местоимения-прилагательные (*ваш, который, мой, самый, какой-то* и т.д.).

3. Структуры, соответствующие существительным.

3.1. В простом предложении могут стоять подряд два существительных, тогда структура будет иметь следующий вид

N	N

.

Предикат здесь будет иметь вид $P(n_1, n_2)$.

3.2. В предложении существительное может согласовываться с наречием. В этом случае имеется структура

N	AdV

и предикат $P(n, adv)$.

Остальные структуры, соответствующие существительным, есть суть одно и то же, что структуры, соответствующие глаголу (кроме случаев 1.5, 1.6) и прилагательному (кроме случая 2.6).

В общем виде проделанное можно изобразить на схеме

предложение ↔ … структура_1 ↔ предикат_1.1 … предикат_1.k … структура_q ↔ предикат_q.1 … предикат_q.k

Предикаты здесь можно брать как определенные только что описанным способом, так и определенные другими способами в разд. 1 и 2.

6. СОПОСТАВЛЕНИЕ ТЕКСТА И ПОТОКОВ

6.1. Формирование потоков

На вход поступает текст, т.е. упорядоченный набор предложений $p_1 p_2 ... p_N$. На выходе формируется несколько потоков:

$$S_1 = < s_{11}, s_{12}, ..., s_{1m_1}, ... >$$

. . . . . .

$$S_k = < s_{k1}, s_{k2}, ..., s_{km_k}, ... >$$

Простейший вспомогательный поток состоит из упорядоченных пар $< 1, p_1, 2, p_2, ..., N, p_N >$, где первая компонента — номер предложения, а вторая — само предложение.

Информацию о словообразовании можно поместить в потоки вида $< h, k_1, L_1, k_2, L_2, ... >$, где h — заголовок потока, например, конкретный суффикс; k_i — номер предложения, где встретилось слово с данным суффиксом (т.е. k_i — номера не всех предложений, а только тех, в которых встречаются эти слова); L_i — список слов с данным суффиксом, содержащихся в данном предложении. Иногда удобнее в поток записывать не сами объекты, а указатели на них, т.е. адреса, где находятся объекты (предложения или слова из предложений). В частности, могут быть указатели на объекты в других потоках, а не в исходном тексте.

С лексическими функциями тоже могут быть ассоциированы потоки, аналогичные потокам, содержащим информацию о словообразовании.

Сопоставленные исходному тексту конечные модели, которые будут в какой-то мере отражать смысловую структуру текста, будем также формировать в виде потоков.

6.2. Формирование основных множеств моделей

Выделяем, например, все существительные из предложений и записываем их в поток: $<1, n_1^1, ..., n_{l_1}^1; 2, n_1^2, ..., n_{l_2}^2; ...>$, где последовательно записываются номера предложений и списки существительных, входящих в данное предложение (l_i — длина списка). Причём номер предложения, в котором нет существительных, может быть пропущен.

С точки зрения программиста, проще работать с потоком, описанным выше. Но с математической точки зрения удобнее будет рассматривать поток, состоящий из пар

$$<<1, n_1^1>, ..., <1, n_{l_1}^1>, <2, n_1^2>, ..., <2, n_{l_2}^2>, ...>.$$

Обозначим через $C = \left\{ < t, n_j^t > | \, t = \overline{1, N}, j = \overline{1, l_t} \right\}$ — множество всех пар, встречающихся в потоке. Основными множествами моделей будут множества вида $C_0 / \sim$, где $C_0 \subseteq C$, $\sim$ — некоторое отношение эквивалентности.

Отношения эквивалентности будут возникать примерно так же, как в конструкции Генцена при доказательстве теоремы о существовании модели [4]. Пары вида $< t, c_j^t > (t = 1, ..., N)$ могут рассматриваться как константы и в зависимости от высказываний об этих константах, некоторые из них мы объявляем эквивалентными.

Аналогично, воспользовавшись полученным потоком, можно будет применить теорему об опускании типов [4] и также в результате получить некоторые модели.

Отметим, что в процессе применения конструкции Генцена на каждом этапе необходимо проверять непротиворечивость соответствующих теорий. При компьютерной обработке текста на естественном языке может быть использована только частичная проверка на непротиворечивость. Например, проверяем, что отношения типа «над» или «под» действительно являются транзитивными; если про какой-то предмет сказано, что он белого цвета, то про него не сказано, что он одновременно черного цвета и т.д.

ЗАКЛЮЧЕНИЕ

Большой интерес с лингвистической точки зрения представляют собой различные модели смысла текста, и более широко — различные подходы к отображению семантики текстов на естественном языке. Поэтому была предпринята попытка исследовать смысл текста, исходя из предварительного структурного разбора этого текста.

В рамках реализуемого проекта предполагалось разработать разнообразные алгоритмы сопоставления предикатов и формул узкого исчисления предикатов для текстов на естественном языке, которые в дальнейшем могут быть подвергнуты изучению и различным преобразованиям средствами математической логики.

В данной работе предложен большой спектр таких предикатов и формул логики первого порядка. Отметим, однако, что пока предложенные предикаты и формулы в первую очередь связаны с грамматической и синтаксической структурой предложений.

Это означает, что несмотря на необходимость данного этапа работы, надо отметить, что пока в полученных формулах недостаточно отражена семантическая структура текста, и необходимо дальнейшее продолжение исследований.

Заметим также, что на данном, по нашему мнению, более простом, этапе был использован большой объем фактической информации из классической и математической лингвистики и математической логики, что говорит о сложности проблемы в целом.

Было разработано также техническое задание, не вошедшее в текст статьи, на разработку системы разнообразного статистического анализа текстов, которое передано программисту высокой квалификации для реализации.

В настоящее время система уже реализована. В качестве данных для исследования были взяты тексты из художественной литературы и из словарных статей толкового словаря, предложения из устной речи, имеющие различное строение.

Для анализа предложений и текстов система использует функции программы Dialing синтаксического и морфологического разбора [7]. С её помощью может быть осуществлён поиск нужного слова в тексте или предложении, вычислена частота встречаемости этого слова, определена лексическая сочетаемость, возможен поиск предложения с заданной структурой по всему тексту, а также определение валентности слова и нахождение его актантов. Эта программа позволяет производить параллельно синтаксический

и морфологический разбор, что является существенным при обработке текстов не только на синтаксическом, но и на семантическом уровне.

СПИСОК ЛИТЕРАТУРЫ

1. **Мельчук И.А.** Опыт теории лингвистических моделей типа «Смысл⇔Текст». — М., 1974. — 315 с.
2. **Апресян Ю.Д.** Экспериментальное исследование семантики русского глагола. — М.: Наука, 1967. — 251 с.
3. **Маркус С.** Теоретико-множественные модели языков. — М.: Наука, 1970. — 332 с.
4. **Сакс Дж. Е.** Теория насыщенных моделей. — М.: Мир, 1976. — 192 с.
5. **Современный** русский язык: Учеб. для филол. спец. высших учебных заведений / Под ред. В.А. Белошапковой. — М.: Азбуковник, 1997. — 928 с.
6. **Современный** русский язык: Учеб. для филол. спец. высших учебных заведений / Под ред. Д.Э. Розенталя. — М.: Изд. МГУ, 1971. — 636 с.
7. **Сокирко А.В.** Семантические словари в автоматической обработке текста // Канд. дисс., МГПИИЯ. — Москва, 2000. — 108 с.

А.А. Винокуров, И.В. Ильин, И.В. Лобив, Ф.А. Мурзин, О.Н. Половинко, Д.Ф. Семич

О НЕКОТОРЫХ ЗАДАЧАХ, СВЯЗАННЫХ С АВТОМАТИЗАЦИЕЙ ПРОЦЕССА ЯДЕРНОГО КАРОТАЖА НЕФТЯНЫХ СКВАЖИН*

ВВЕДЕНИЕ

Метод ядерного каротажа [1–6] состоит в том, что в скважину на тросе опускается источник нейтронов. При их излучении они взаимодействуют с внешней средой, в которой происходят ядерные реакции, вследствие которых образуются осколки, представляющие собой отклик внешней среды. В самом приборе установлены специальные кристаллы (как правило, это соединения редкоземельных металлов), и при воздействии различных частиц в них происходят вспышки (флюоресценция). Эти вспышки регистрируются фотоэлементами — так возникают энергетические спектры.

На первом этапе обработки энергетических спектров необходимо произвести энергетическую привязку. По характерным фотопикам произвести интерполяцию, при которой ось абсцисс каналов (амплитуда зарегистрированных сигналов в единицах АЦП) отображается в ось энергий путем линейной интерполяции. Все последующие расчеты выполняются, исходя из пары ось абсцисс — ось энергий, и не требуют вмешательства оператора.

Сложность этой процедуры состоит в том, что данный процесс имеет большой субъективный фактор. Интерпретатор должен хорошо знать взаимодействие нейтронов с веществом как быстрых нейтронов, так и тепловых. Он должен уметь различать характерные фотопики на спектрах неупругого рассеяния, радиационного захвата и наведенной активности. Он также должен знать энергию гамма-квантов естественной радиоактивности (ряд урана, ряд тория, калий), правильно их интерпретировать.

Все это накладывает большие требования на знания в ядерной физике и геофизике и на опыт при обработке энергетических спектров. При этом правильность выполнения энергетической привязки может иметь погрешность в силу субъективных и технических причин, изменения спектра при температуре, низкой статистики высокоэнергетических гамма-квантов.

Поэтому автоматизация технологий ядерного каротажа является актуальной задачей.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

Каротаж, основанный на измерении естественной радиоактивности, проводится следующим образом. Сначала прибор калибруют на образцах, содержащих уран, радиоактивный изотоп калия и торий. Кроме того, берётся фоновый образец — цемент. Потом откалиброванным прибором можно измерить содержание этих элементов в породе (по глубине).

В литологии расклассифицированы породы, т.е. по отношениям U/K, K/Th, Th/U можно опознать глины разных видов, полевой шпат, слюду, каолин, боксит, мергель, монтморрилонит и многое другое, т.е. в итоге можно произвести литологическое расслоение среды по глубине.

В данной статье описываются математические постановки нескольких задач, возникающих при автоматизации процесса ядерного каротажа, и указаны методы их решения.

1. ЭНЕРГЕТИЧЕСКАЯ ПРИВЯЗКА СПЕКТРОВ

На первом этапе обработки энергетических спектров необходимо произвести энергетическую привязку. Алгоритм, который будет описан ниже, повышает точность энергетической привязки и устраняет субъективный фактор при решении данной задачи.

1.1. Входные данные

Входными данными являются энергетические спектры различных видов. Типичный энергетический спектр приведен ниже.

Рис.1. Пример энергетического спектра

По оси абсцисс расположены каналы, или энергии вспышек. По оси ординат откладывается количество зарегистрированных вспышек на данном интервале времени. Для большей наглядности, начиная со 180-го канала, амплитуда увеличена в 10 раз.

На графике видны фотопики, соответствующие отдельным элементам. Некоторым элементам соответствуют несколько фотопиков. Например, железу соответствуют три фотопика. Два из них показаны на графике. Третий, находящийся левее данных двух, не показан. Он почти не проявляется в силу наличия шумов.

1.2. Выделение фотопиков элементов

Предполагаем, что в нашем распоряжении имеется список элементов $El = \{e_1, \ldots, e_N\}$, например, $El = \{B, H, O, Fe, Si, Ca, Cl, C\}$.

При отсутствии теплового дрейфа прибора, точнее при фиксированной температуре, каждому элементу однозначно соответствует набор каналов, на которых данный элемент может быть зарегистрирован.

Обозначим $\bar{x}(e_i) = < x_1(e_i), \ldots, x_{K_i}(e_i) >$ набор каналов, соответствующий элементу e_i. Например, железу соответствует тройка

$$\bar{x}(Fe) = < x_1(Fe), x_2(Fe), x_3(Fe) > .$$

Вследствие теплового дрейфа и других шумов данные координаты, на которых осуществляется контроль на фотопики, могут незначительно смещаться по оси каналов.

В настоящее время энергетическая привязка осуществляется следующим образом.

1. Щелчками мыши на экране пользователь выделяет пики бора и железа. При этом бор выделяется на ГИНР-е, а железо на ГИРЗ-е. ГИНР — спектр гамма-излучения наведённой радиоактивности, и ГИРЗ — спектр гамма-излучения радиационного захвата.

2. Используя соответствующие энергии элементов, получим две точки на плоскости «каналы-энергии». Проводим через них прямую, отображающую ось каналов в ось энергий.

3. Предположим, что уравнение упомянутой выше прямой имеет вид $y = ax + b$. Тогда для любого элемента e, взяв табличное значение энергии $E(e)$, соответствующее данному элементу, можно вычислить номер канала, на котором он должен находиться $x(e) = (E(e) - b)/\alpha$.

4. Далее отыскиваем фотопики на графике спектра. Это можно делать различными способами.

- Можно искать локальные максимумы в окрестности $x(e)$. Размер окрестности задаётся пользователем.
- Можно взять с некоторого спектра маленький фрагмент графика вблизи $x(e)$, иначе говоря, образец фотопика, если он достаточно проявлен.
- Потом можно искать фотопик на другом графике с помощью корреляционной функции с данным образцом.

Если в данной окрестности локальный максимум находится в одной точке, а всплеск корреляционной функции в другой, то мы отдаём предпочтение второму варианту.

Наконец, корреляционная функция может дать два всплеска одинаковой величины. Это бывает для *H*(*E*=2230 кэВ). В этом случае отдаём предпочтение правому всплеску. Считаем, что левый всплеск индуцирует $C_0(E = 1330$ кэВ).

5. В итоге элемент e может быть локализован в результате упомянутой выше процедуры в некоторой точке $x'(e) \neq x(e)$.

Отметим, однако, что $x'(e)$ и $x(e)$ достаточно близки, что обусловлено особенностями протекающих физических процессов.

6. Предположим теперь, что $x'(e_1) \ldots x'(e_k)$ — координаты всех найденных фотопиков на оси каналов.

Далее пусть $x(e_1) \ldots x(e_k)$ — точки, в которых они должны находиться в соответствии с п. 2.

Производим растяжение или сжатие спектра так, чтобы точки $x'(e_i)$ перешли в точки $x(e_i)$ соответственно.

Заметим, что интервалы $[\, x'(e_i)\,;\, x'(e_{i+1})\,]$, $[\, x(e_i)\,;\, x(e_{i+1})\,]$ могут содержать различное количество точек. Поэтому необходимо использовать линейную интерполяцию, чтобы отобразить один интервал на другой, т.к. целым узлам на отрезке $[\, x(e_i)\,;\, x(e_{i+1})\,]$ могут соответствовать нецелые узлы на отрезке $[\, x'(e_i)\,;\, x'(e_{i+1})\,]$. Можно высказать предположение, что таким образом мы компенсируем тепловой дрейф прибора.

2. ОБРАБОТКА ВРЕМЕННЫХ СПЕКТРОВ

2.1. Описание основного алгоритма

Известно, что с точностью до шумов небольшой амплитуды временной спектр состоит из трёх частей, характеризующих отклик от ближней, средней и дальней зон.

Обозначим временной спектр $T(i)$, где i — номер отсчёта по времени, в нашем случае $1 \le i \le 64$. В типичных случаях существуют точки $i_0 < i_1 < i_2 < i_3$ такие, что на каждом из интервалов $I_k = [i_{k-1}, i_k]$, $k = 1,2,3$ спектр представляет собой экспоненту вида $f_k(x) = e^{-\lambda_k (x + x_0^k)}$, $k = 1,2,3$.

Далее можно считать, что $i_0 = 4$. При $i = 1,2,3$ мы имеем неустойчивость работы аппаратуры.

Известно также, что в типичных случаях i_1, i_2, i_3 находятся в окрестностях точек 10, 20, 30 соответственно. Значения $T(i)$ при $i > 35$ не представляют интереса.

Для того чтобы локализовать i_1, i_2, используется дискретное вейвлет-преобразование, построенное на основе вейвлета Добеши DB4.

На спектре $S(i)$ вейвлет-преобразования в окрестностях точек 10 и 20 имеются пики большой амплитуды. Они локализуют место изменения параметра λ, т. е. i_1, i_2.

Более того, полагаем i_1 равным такому i, что при $8 \le i \le 12$ и $|S(i)|$ достигаем максимума в указанной окрестности. Заметим, что во всех примерах, которые были просчитаны, такая точка была только одна. Точку i_2 выбираем аналогично на интервале $27 \le i \le 33$.

Вблизи точки 30 на спектре также имеются два локальных максимума небольшой амплитуды. Часто они имеют разные знаки.

Детальный анализ показывает, что они не характеризуют место изменения параметра λ. Их появление вызвано тем, что более высокочастотные гармоники «загашивают» неточность приближения исходного сигнала линейной комбинацией низкочастотных гармоник. Поэтому для нахождения i_3 используется другой метод.

Сначала подсчитаем суммы $\sum_k = \sum_{j=k}^{64} T(j)$, $10 \le k \le 64$.

Легко видеть, что $\sum_{10} \geq \sum_{11} \geq \cdots \geq \sum_{64}$. В программной реализации указанные суммы подсчитываются в обратном порядке.

Пусть $M = \sum_{10}$, C — некоторая константа, мы взяли $C = 1/10$. Находим такое k, что $\sum_{k-1} > C \cdot M \geq \sum_{k}$.

Если $k > i_2$, то полагаем $i_3 = k$. В противном случае полагаем $i_2 = k$ и считаем, что спектр распадается не на три, а на два участка $[i_0, i_1]$, $[i_1, i_2]$. Отметим ещё, что k не может быть расположено левее i_1, так как там значения $T(i)$ большие, а вся описанная выше технология направлена на «отсечение» хвоста, почти сплошь состоящего из нулей.

Дальнейшая обработка состоит в следующем. Переходим к спектру, определённому не на номерах каналов (т.е. на номерах временных отсчётов), а к спектру, определённому в реальном времени в микросекундах, полагая

$$\mathbf{T}(x_i) = T(i), \; x_i = 60 + i \cdot 40 .$$

Имеются три соответствующих множества:

$$X_1 = \{x_i : i_0 \leq i \leq i_1\};$$
$$X_2 = \{x_i : i_1 \leq i \leq i_2\};$$
$$X_3 = \{x_i : i_2 \leq i \leq i_3\}.$$

На каждом из участков

$$\mathbf{T}(x_i) \approx e^{\lambda_k (x + x_0^s)}, \; k = 1,2,3.$$

Поэтому параметры экспонент можно вычислить, используя логарифмирование и метод наименьших квадратов для линейной функции.

Нас интересуют величины $\tau_k = 1/\lambda_k$, $k = 1,2,3$, характеризующие времена жизни гамма-квантов, приходящих из ближней, средней и дальней зон.

В случае, когда средняя зона вырождается, сливаясь с ближней, получаем два значения τ_1, τ_2.

В алгоритме предусмотрена возможность усреднения значений λ и соответственно τ с помощью медианной фильтрации.

Фиксируем небольшое h, обычно $h = 3, 4, 5$.

Полагаем $i_0 = 4$, i_1 находим при помощи вейвлет-преобразования. Далее рассмотрим отрезки вида $[i_0, i]$, $i_1 - h \leq i \leq i_1 + h$.

Для каждого такого отрезка подсчитываем своё λ, заметим, что их нечётное число. Полагаем λ_1 равным медиане из полученных значений λ. Это означает, что если все данные λ упорядочим по возрастанию, то h штук будет $\le \lambda_1$ и h штук $\ge \lambda_1$. Именно так работает медианный фильтр, подавляющий импульсные помехи.

Соответственно корректируем i_1, выбирая именно то i, которому соответствует данное медианное. На следующем участке действуем аналогично. Мы имеем новое i_1 и рассматриваем отрезки вида $[i_2, i]$, $i_2 - h \le i \le i_2 + h$.

2.2. Метод наименьших квадратов

Предположим, что задана некоторая функция $y(x)$, $f(x, a_0, a_1, \cdots, a_k)$ — её приближение.

Для конечного набора точек можно написать $y_i = f(x_i, a_0, a_1, \cdots, a_k) + \xi_i$, где ξ_i — независимая случайная величина.

Минимизируем функционал, выражающий среднеквадратичную погрешность $\Phi(a_0, \cdots, a_k) = \sum_i (f(x_i, a_0, a_1, \cdots, a_k) - y_i)^2$.

В случае, когда приближающая функция линейная

$$f(x) = a_0 + a_1 x = a + bx,\ a_0 = a,\ a_1 = b,$$

минимизируем

$$\Phi(a_0, a_1) = \sum_i (a_0 + a_1 x_i - y_i)^2 .$$

Приравнивая производные нулю, получаем

$$\frac{\partial \Phi}{\partial a_0} = 0;\ \frac{\partial \Phi}{\partial a_1} = 0 .$$

Известно, что полученные уравнения допускают явное решение

$$b = a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2},$$

$$a = a_0 = \frac{1}{n}(\sum_{i=1}^{n} y_i - a_1 \cdot \sum_{i=1}^{n} x_i).$$

2.3. Дискретное вейвлет-преобразование на основе DB4

2.3.1. Рекуррентно определяем набор функций $\varphi_j(x)$, которые называются масштабирующими функциями $\varphi_j = \sqrt{2}\sum_{k=0}^{2M-1} h_k \varphi_{j-1}(2x-k)$.

Для DB4 полагаем

$$\varphi_0(x) = \varphi_H(x) = \begin{cases} 1, & 0 < x < 1; \\ 0, & \textit{в противном случае.} \end{cases}$$

2.3.2. Базисные функции задаются посредством формулы

$$\psi_j(x) = \sqrt{2}\sum_{k=0}^{2M-1} g_k \varphi_j(2x-k),$$

$$g_k = (-1)^k h_{2M-k-1}.$$

Для DB4 $M = 2$ и чаще всего используются коэффициенты

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}},\ h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}},$$

$$h_3 = \frac{3-\sqrt{3}}{4\sqrt{2}},\ h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

2.3.3. В дискретном случае возникает матрица размерности 64 × 64, строчки которой являются базисными векторами. Дискретное вейвлет-преобразование принимает вид умножения матрицы на вектор. Выбор базиса неоднозначен.

Мы используем более-менее стандартный метод построения базиса. Применяем сжатие вдоль оси x в 2, 4, 8 и так далее раз и параллельные переносы вдоль той же оси на число отсчётов кратное 32, 16, 8, 4, 2 в зависимости от того, насколько велик носитель данной гармоники. Иначе говоря, гармоники выбираются по аналогии с преобразованием Хаара.

3. ВЫЧИСЛЕНИЕ ЧИСТЫХ СПЕКТРОВ

В процессе работы прибора получаются четыре сигнала

$$V_1(i),\ V_2(i),\ V_3(i),\ V_4(i),\ 1 \le i \le 512.$$

Гамма-излучение радиоактивного захвата: ГИРЗ$= V_1 + V_3 + V_4$.

Гамма-излучение наведённой радиоактивности: ГИНР$= V_2 - CV_3$, где V_3 — гамма-излучение фона.

Обычно константу C выбирают опытным путём. Критерием является следующее. Необходимо в разности $V_2 - CV_3$ подавить пик водорода на 135 канале. Эксперименты показывают, что, как правило, $0.85 \le C \le 1.1$. В отдельных случаях константа C может быть меньше. В одном из тестов получилось, что $C = 0.65$.

Заметим, что на обоих спектрах V_2 и V_3 присутствуют пики на 135-ом канале. При этом на V_3 пик немного выше и шире, чем на V_2, если измерять их высоту относительно основных несущих экспонент соответствующих спектров.

Сначала был предложен следующий алгоритм. Прежде всего вычисляем параметры несущих экспонент. Для этого необходимо взять на графиках несколько точек. Мы берём точки $i_1 = 128$, $i_2 = 129$, $i_3 = 130$, $i_4 = 140$, $i_5 = 141$, $i_6 = 142$. Эти точки лежат вне области, где расположен пик водорода. А именно, окрестность точки 135 вырезана. Более того, левее 128 достаточно близко находится пик цезия. Мы его также не задеваем.

С точностью до небольших пиков спектры V_2, V_3, аналогично временным спектрам, представляют собой функции вида $e^{-\lambda(x+x_0)}$, $\lambda > 0$. Соответственно, после логарифмирования получаем линейную функцию

$$f(x) = \ln(e^{-\lambda(x+x_0)}) = -\lambda(x + x_0) = a_0 + a_1 x,$$
$$a_0 = -\lambda x_0,\ a_1 = -\lambda.$$

Поэтому параметры экспонент можно вычислить, используя логарифмирование и метод наименьших квадратов.

Полагаем $y_k^s = \ln(V_s(i_k))$, $1 \le k \le 6$; $s = 2,3$.

Далее применяем метод наименьших квадратов к наборам точек

$$\{< i_1, y_1^2 >, \ldots, < i_6, y_6^2 >\},$$
$$\{< i_1, y_1^3 >, \ldots, < i_6, y_6^3 >\},$$

получаем, соответственно, уравнения двух прямых

$$l_2(x) = a_0^2 + a_1^2 x,$$
$$l_3(x) = a_0^3 + a_1^3 x.$$

Далее пусть $L_s(x) = \exp(l_s(x)) = e^{a_0^s + a_1^s x}$, $s = 2,3$.

Обозначим теперь $l_2 = L_2(135)$, $l_3 = L_3(135)$.

Иначе говоря, произведена экстраполяция экспоненты внутрь области, на которой расположен пик, а пик при этом игнорируется.

Кроме того, имеются величины $v_2 = V_2(135)$, $v_3 = V_3(135)$.

Полагаем $C = \dfrac{v_2 - v_3}{l_2 - l_3}$.

Потом мы отказались от работы в каналах и перешли к работе в энергиях. Энергия водорода равна 2230 кэВ. Шаг дискретизации равен 10 кэВ. Поэтому после энергетической привязки к полученным спектрам (в энергиях) можно применить аналогичный алгоритм в точке $i = 223$.

Было выяснено, что при работе в энергиях константа вычета фона вычисляется лучше. Однако оказалось, что для малых значений константы (в частности для теста, когда $C = 0.65$) метод дает завышенные значения константы, если при экстраполяции экспоненты взять шесть точек слева и шесть справа от пика.

Это означает, что в действительности данный участок графика недостаточно хорошо приближается экспонентой. Попытались варьировать количество точек справа и слева от области пика. В результате численных экспериментов было установлено, что если справа взять большое количество точек, около ста или даже более, а слева не брать вообще, то качество вычисления константы вычета фона оказывается очень высоким.

4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Еще раз сформулируем задачи, которые решаются в процессе обработки информации, полученной при ядерном каротаже. На первом этапе обработки энергетических спектров необходимо произвести энергетическую при-

вязку. По характерным фотопикам произвести интерполяцию, при которой ось абсцисс каналов (амплитуда зарегистрированных сигналов в единицах АЦП) отображается в ось энергий путем линейной интерполяции.

Известно, что временной спектр с точностью до шумов небольшой амплитуды состоит из трёх частей, которые достаточно точно описываются некоторыми экспонентами, характеризующими отклик от ближней, средней и дальней зон. Необходимо выделить эти участки и определить параметры, задающие данную экспоненту.

Третья задача заключается в подборе константы, с помощью которой можно было бы подавлять всплеск, отображающий водород на втором сигнале (из четырёх получаемых) с помощью вычитания из него третьего сигнала, умноженного на получаемую константу. Данная константа называется константой вычета фона.

В результате проведенных исследований были разработаны алгоритмы и создано программное обеспечение для автоматизации процесса ИНГК — импульсного нейтронного гамма каротажа. Программа позволяет обрабатывать данные как энергетических спектров, так и временных.

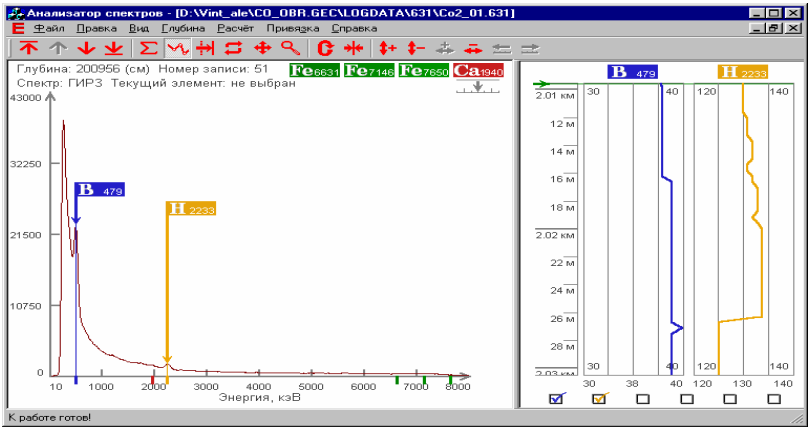


Рис. 2. Главное окно программы SpectrumAnalyzer

В настоящее время возможности программы расширены. Ее можно применять для обработки данных, полученных в процессе гамма-спектроскопии естественной радиоактивности, и для обработки временных спектров в соответствии с водородокомпенсирующей технологией.

СПИСОК ЛИТЕРАТУРЫ

1. **Новиков Г.Ф., Копков Ю.Н.** Радиоактивные методы разведки // Л.: «Недра», 1965. — 755 с.
2. **Хуснуллин М.Х.** Геофизические методы контроля разработки нефтяных пластов // М.: «Недра», 1989. — 188 с.
3. **Резванов Р.А.** Радиоактивные и другие неэлектрические методы исследования скважин // М.: «Недра», 1982. — 367 с.
4. **Алексеев Ф.А., Головацкая И.В., Гулин Ю.А. и др.** Ядерная геофизика при исследовании нефтяных месторождений // М.: «Недра», 1978. — 359 с.
5. **Фертл В.Х.** Спектрометрия естественного гамма-излучения в скважине // Нефть, газ и нефтехимия за рубежом. — 1986. — № 3—11.
6. **Kozhevnikov D.A., Shagin V.L.** A method of treating the spectral response of a tool in open and cased boreholes to determine the natural radioactivity of rocks // Nucl. Geophys. — 1989. — Vol.3, №. 1. — P. 17—29.

Т. А. Волянская

ВИРТУАЛЬНЫЙ МУЗЕЙ ИСТОРИИ ИНФОРМАТИКИ В СИБИРИ: МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ И МОДЕЛЬ ПОЛЬЗОВАТЕЛЯ*

ВВЕДЕНИЕ

Статья посвящена проекту создания виртуального музея истории информатики в Сибири, работа над которым ведется коллективом сотрудников ИСИ СО РАН, ИМ СО РАН и НГУ при финансовой поддержке Российского гуманитарного научного фонда (грант РГНФ N 02-05-12010). Музей создается в виде информационно-поисковой, справочной адаптивной гипермедиа-системы, доступной в интернете [1–5].

В настоящее время в интернете представлено большое число виртуальных музеев, подавляющее большинство из которых реализовано с использованием традиционных гипермедиа-технологий. Одним из ограничений традиционных гипермедиа-систем, предназначенных для использования различными категориями пользователей, является то, что они предоставляют одно и то же информационное содержание и один и тот же механизм навигации всем пользователям. Разрабатываемый виртуальный музей предназначен для различных категорий пользователей, и его посетители, имеющие различные цели, интересы, знания и предпочтения, могут нуждаться в различных частях содержащейся информации и использовать различные пути для навигации. Поэтому при создании музея особое внимание уделяется вопросам его адаптации [2, 3].

Структура статьи следующая: в первом разделе кратко представлены адаптивная гипермедиа и адаптивные гипермедиа-системы. Во втором разделе рассматриваются архитектура адаптивных гипермедиа-систем и ее основные компоненты (модель предметной области, модель пользователя и модель адаптации), а также вопросы моделирования предметной области и моделирования пользователя. Третий раздел посвящен вопросам, относящимся к виртуальному музею истории информатики в Сибири: рассматривается структура и содержимое музея, категории пользователей, а также представление знаний предметной области и моделирование пользователя.

* Работа выполнена при финансовой поддержке Российского гуманитарного научного фонда (грант № 02-05-12010) и Министерства образования РФ.

1. АДАПТИВНЫЕ ГИПЕРМЕДИА-СИСТЕМЫ

Адаптивная гипермедиа (Adaptive Hypermedia) (АГ) — альтернатива традиционному подходу разработки гипермедиа-систем. *Адаптивные гипермедиа-системы* (АГС) поддерживают модель целей, знаний, интересов, предпочтений и других особенностей индивидуального пользователя и используют ее в течение взаимодействия для адаптации к потребностям пользователя [3,6–10].

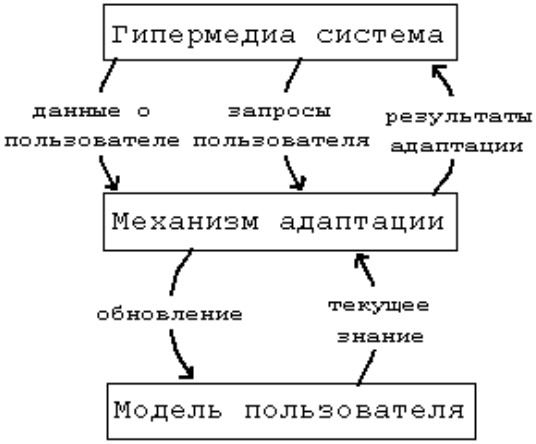


Рис. 1. Адаптивная гипермедиа-система

АГС в общем случае поддерживают следующие три вида адаптации: к данным пользователя, к рабочим характеристикам и к данным окружения. Данные пользователя включают различные характеристики пользователей: знания, цели, подготовку, опыт в гиперпространстве, предпочтения, интересы и индивидуальные особенности пользователя. Рабочие характеристики включают данные о взаимодействии пользователя с системой, которые не могут быть сведены к характеристикам пользователя, но все еще могут использоваться для принятия решений адаптации. Данные окружения включают все аспекты пользовательского окружения, которые не связаны с пользователями (например, аппаратные средства, программное обеспечение, пропускная способность сети или местонахождение пользователя) [3, 6–8].

Рис. 2. Классификация технологий АГ

АГС системы обеспечивают *адаптивное представление (adaptive presentation),* т.е. адаптацию содержания гипердокументов, и *адаптивную поддержку навигации (adaptive navigation support)*, т. е. адаптацию структуры гиперссылок. Смысл методов адаптивного представления состоит в том, чтобы адаптировать содержание страницы, к которой обращается отдельный пользователь, к текущему знанию, предпочтениям, интересам, целям и другим характеристикам пользователя. Основные методы адаптивного представления текста — это дополнительные, предварительные и сравнительные объяснения, варианты объяснения и сортировка. Следующие технологии используются для реализации вышеперечисленных методов адаптивного представления текста: условный текст, стрейч-текст, варианты фрагментов и варианты страниц, фреймовая технология. Смысл методов адаптивной навигационной поддержки состоит в том, чтобы помочь пользователям найти путь в гиперпространстве с помощью адаптации способа представления ссылок к целям, знанию и другим характеристикам индивидуального пользователя. Методы адаптивной навигационной поддержки используются для достижения нескольких целей адаптации: обеспечить

глобальное руководство, локальное руководство, глобальную ориентацию, локальную ориентацию и управление индивидуализированными представлениями в информационных пространствах. Для реализации этих методов применяются следующие технологии: полное руководство, сортировка, сокрытие, аннотирование, генерирование ссылок и адаптация карты [3, 6–8, 10, 11].

2. АРХИТЕКТУРА АГС

На абстрактном уровне АГС можно описать с помощью следующей архитектуры, базирующейся на трех компонентах [2,11,12,15–17]:

- модель предметной области (Domain Model),
- модель пользователя (User Model),
- модель адаптации (Adaptation Model).

2.1. Модель предметной области (МО)

Модель предметной области (МО) дает описание предметной области (ПО) на концептуальном уровне и представляет собой совокупность концептов и межконцептных отношений. *Концепты* — это абстрактные объекты, используемые для представления элементов информации ПО. Существуют концепты нижнего уровня или *атомные концепты*, которые соответствуют одному фрагменту информации, и концепты высшего уровня или *составные концепты*, которые состоят из множества других концептов (высшего или нижнего уровней) [11,12,15–17].

Межконцептные отношения представляют различные отношения между концептами. Следующие типы отношений между концептами являются наиболее используемыми:

- тип *часть (part-of)* представляет композиционное отношение: показывает, какую долю первый концепт составляет от второго;
- тип *связь (link)* означает, что существует связь первого концепта со вторым (например, в виде гиперссылки);
- тип *предпосылка (prerequisite)* означает, что первый концепт должен быть предварительно изучен до изучения второго концепта;
- тип *ингибитор (inhibitor)* означает, что первый концепт не должен изучаться до тех пор, пока не будет изучен второй.

Можно выделить три типа моделей предметной области, различающихся по уровню сложности структуры [7]:

- самую простую структуру имеет МО *первого уровня*, являющаяся независимым множеством концептов (отсутствуют межконцептные отношения);
- МО *второго уровня* (сетевая МО) предполагает наличие связей между концептами и представляет собой семантическую сеть, состоящую из концептов и межконцептных отношений. Может существовать несколько типов концептов и межконцептных отношений;
- МО *третьего уровня* (фреймовая МО) предполагает наличие у концептов внутренней структуры в виде множества атрибутов, при этом концепты различных типов могут иметь различные множества атрибутов.

Одной из самых важных функций МО является обеспечение структуры для представления знаний пользователя ПО.

АГС можно подразделить на три основные группы в зависимости от метода организации связи между МО (концептами) и гиперпространством системы (гипермедиа-страницами) [7]:

- самый простой метод — это *индексация страниц* концептами, относящимися к содержимому этих страниц; он может применяться даже для МО первого уровня;
- второй метод, похожий на предыдущий, — это *индексация фрагментов*: содержимое страницы разбивается на множество фрагментов, каждый из которых отдельно индексируется множеством концептов, относящихся к содержимому данного фрагмента. При разбиении на относительно небольшие фрагменты каждый фрагмент индексируется ровно одним концептом. Этот метод также можно использовать при МО первого уровня;
- третий метод (*прямая связь*) отличается от предыдущих методов тем, что для страниц не поддерживаются индексы; гиперпространство АГС строится непосредственно исходя из структуры МО. Таким образом, каждый концепт МО представлен гипермедиа-страницей или гипердокументом, а отношения между концептами соответствуют гиперссылкам между страницами. Страница или документ, представляющий концепт, могут быть как статическими, так и динамическими: т.е. генерироваться налету, исходя из внутренней структуры концепта. Последний метод является самым мощным из всех вышеперечисленных, однако он требует использования МО второго, а лучше третьего уровня.

2.2. Модель пользователя (МП)

2.2.1. Моделирование пользователя

Моделирование пользователя можно определить как адаптацию поведения системы в процессе взаимодействия с ним в соответствии с теми предположениями, которые делает система, основываясь на собранной о пользователе информации. Для обеспечения адаптации АГС необходима модель пользователя, с помощью которой система может определить что, когда и как должно быть адаптировано. По существу МП — это набор фактов о пользователе, сопутствуемый набором процессов, которые ими управляют: добавление в МП новой информации по мере ее поступления, разрешение противоречий в сделанных предположениях и выведение по возможности новой информации. Исследования в области моделирования пользователя затрагивают проблемы представления МП в системе, формирования предположений о пользователе из входных данных или других источников, а также использования системой этих предположений для адаптации взаимодействия к пользователю.

2.2.2. Модель пользователя

Модель пользователя (МП) в АГС предполагает *явное* представление знаний, предпочтений, целей, интересов, истории навигации и других характеристик пользователя и служит для адаптации к пользователю различных аспектов АГС. МП состоит из именованных элементов, для которых хранится набор пар вида атрибут-значение (компонентов МП). На концептуальном уровне можно представлять МП в виде табличной структуры, в которой для каждого элемента хранятся значения атрибутов. Большинство элементов в МП представляют концепты МО. Некоторые другие элементы могут представлять различные аспекты пользователя, такие как цели, предпочтения, интересы или стереотипная классификация (типа новичок, эксперт) и т.д. [11, 12, 15–17].

Можно классифицировать МП согласно некоторым основным аспектам:

– *способ получения (явный и неявный):* при явном способе система в явном виде запрашивает у пользователя информацию, необходимую для создания и обновления модели. Альтернативный подход, при котором система незаметно и ненавязчиво для пользователя конструирует МП, отслеживая и анализируя взаимодействия с пользователем (например, историю навигации), с последующим выведением различных предположений о пользователе;

- *степень специализации (общие и индивидуальные модели):* общая модель предполагает наличие одной общей модели для всех пользователей системы, она используется в случае, когда система предназначена для эксплуатации однородной группой пользователей. Индивидуальная модель принимает во внимание индивидуальные особенности пользователей и поддерживается для каждого пользователя своя. Стереотипная модель является способом комбинирования двух вышеперечисленных моделей: стереотип — набор характеристик, связанных друг с другом, определяющий некоторый класс пользователей;
- *модифицируемость (статические и динамические модели):* в то время как статическая модель не изменяется, динамическая модель постоянно обновляется по мере получения новой информации в течение сеанса взаимодействия с пользователем;
- *временная протяженность (краткосрочные и долгосрочные модели):* в отличие от краткосрочных моделей, долгосрочные модели сохраняются от одного сеанса взаимодействия с пользователем до другого;
- *метод использования (дескриптивные и прескриптивные модели):* более традиционным является дескриптивное использование МП, в этом представлении модель пользователя — простая база данных, содержащая информацию о пользователе, у которой система может запрашивать текущие данные о пользователе. Прескриптивное использование модели предполагает, что система моделирует (имитирует) пользователя для проверки интерпретации его ответа.

2.2.3. Методы моделирования

Можно выделить два основных метода моделирования пользователя: оверлейное и стереотипное моделирование [13, 14].

Оверлейное моделирование (overlay modeling) чаще всего используется в интеллектуальных системах обучения для моделирования знаний, при этом знания пользователя описываются как подмножество знаний ПО эксперта (т.н. «оверлей»). Для каждого концепта МО в модели знаний пользователя вычисляется и сохраняется некоторое значение (или несколько значений), оценивающее уровень знания пользователем этого концепта (оверлейная модель). Оверлейная модель знаний (overlay model) может быть представлена множеством пар «концепт —значения атрибутов».

Следующие атрибуты обычно используются для обозначения уровня знаний концепта пользователем [15–17].

- *«Значение знания»* (или просто *значение*) показывает уровень знания концепта пользователем. Все пары концепт–значение вместе формируют оверлейную модель, которая представляет «знания» пользователя. Некоторые АГС используют булеву модель пользователя, в которой для каждого концепта определено, знает или не знает пользователь концепт. Другие АГС используют небольшой набор значений, например, «неизвестен», «изучен», «хорошо изучен» и «известен» или большой набор (например, проценты).
- Атрибут *«чтения»* показывает, просматривал (читал, изучал) ли пользователь какую-нибудь информацию (фрагмент, страницу или набор страниц) о концепте. В веб-системах атрибут чтения используется, чтобы генерировать различное представление для гиперссылок к посещенным и непосещенным страницам (по умолчанию это фиолетовый и синий цвета). Атрибут чтения может иметь булевы значения в некоторых АГС, в то время как в других АГС это может быть список времен доступа.
- Менее распространенный атрибут — это *«готовность для чтения»*, который показывает, готов ли пользователь для просмотра и изучения информации об этом концепте (это означает, например, что пользователь уже приобрел достаточное количество предварительно необходимых знаний.)

Использование оверлейной модели особенно целесообразно, когда материал обучения может быть представлен в виде иерархии предварительных условий. Как уже было сказано выше, в рамках модели предполагается, что знание пользователя составляет некоторое подмножество знания эксперта, и цель обучения состоит в расширении этого подмножества. Модель также предполагает, что пользователь не будет изучать того, чего не знает эксперт. В частности, не принимаются во внимание неправильные представления и заблуждения, изначально имеющиеся у пользователя, или приобретенные им в процессе обучения. Второй недостаток оверлейной модели заключается в том, что нет механизма для разграничения знаний, которые пользователь еще не приобрел, и знаний, которые еще не были ему представлены, что имеет смысл для стратегии обучения.

Дифференциальная модель (differential model) обращается к этим вопросам. Дифференциальная модель является расширением оверлейной модели. Знание разделено на то, которое было продемонстрировано пользователю, и то, которым пользователь не обладает. Модель подразделяет все знание ПО

на представленное и не представленное пользователю. Оверлейная модель применяется к знанию, уже представленному пользователю. В отличие от оверлейной, дифференциальная модель принимает во внимание неправильные представления и ошибки пользователя.

Пертурбационная модель (perturbation model) принимает во внимание знания, которыми обладает пользователь, но которые не представлены в модели знания эксперта ПО. Пертурбационная модель расширяет модель эксперта добавлением библиотеки ошибок (bug library). Процесс ее создания может быть перечисляющим или порождающим. Перечисляющий процесс составляет список всех возможных неправильных представлений с помощью анализа ПО и ошибок, которые допускает пользователь. Порождающий метод пытается генерировать ошибки исходя из лежащей в основе познавательной теории. Оверлейная модель может быть применена поверх комбинированной модели эксперта и библиотеки ошибок. Как и для простой оверлейной модели, цель обучения — увеличить подмножество знания эксперта при исключении неправильных представлений.

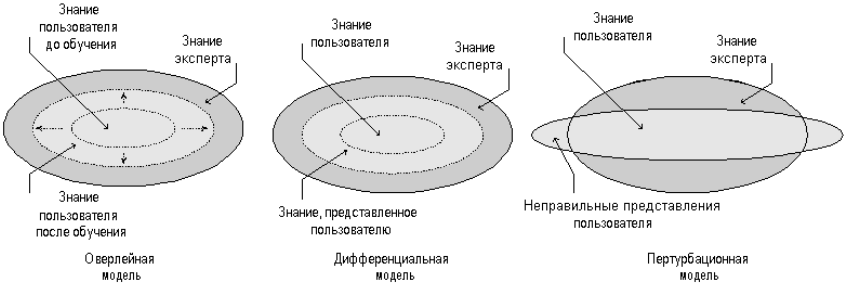


Рис. 3. Оверлейная, дифференциальная и пертурбационная модели знаний

Стереотипное моделирование (stereotype modeling) — один из первых методов в области моделирования пользователя, предложенный в 1979 году Элэйн Рич (Elaine Rich). Этот метод целесообразно использовать в случаях, когда требуется быстрая, но не обязательно правильная оценка пользователя.

Можно определить стереотип как набор некоторых взаимосвязанных характеристик, присущих всем членам определенной подгруппы пользователей. Стереотипная модель различает несколько типичных или «стереотипных» пользователей (например, «новичок», «средний», «эксперт» и т.п.). В процессе моделирования все пользователи группируются в классы (каждый пользователь ассоциируется с одним или более стереотипом), при этом

полагается, что пользователи, принадлежащие к одному классу, имеют одинаковые характеристики. Стереотипная классификация может быть сделана для каждого аспекта адаптации. Стереотипная модель может быть представлена как множество пар «стереотип–значение», где значением может быть «истина» или «ложь» (означающее, принадлежит ли пользователь данному стереотипу).

Кроме характеристик подгрупп пользователей, стереотипы могут содержать так называемые триггеры (инициирующие условия), которые представляют ключевые характеристики, позволяющие идентифицировать пользователя как принадлежащего к соответствующей подгруппе пользователей. Триггеры могут относиться к текущим пользовательским характеристикам, эксплуатационным характеристикам (например, истории навигации), данным окружения (например, данным о конфигурации, оборудовании). Стереотипы применяются к пользователю, если они назначены «вручную» или если их триггеры совпадают с доступной информацией о пользователе (автоматическая классификация). В результате все характеристики соответствующего стереотипа приписываются пользователю.

Суммируя вышесказанное, можно представить, что стереотип состоит из следующих частей:

- множества инициирующих условий (триггеров), являющиеся логическими выражениями, активирующими стереотип;
- множества условий отвода (ретракций), которые ответственны за деактивацию активного стереотипа;
- множества предположений (выводов) стереотипа, которые служат предположениями по умолчанию при ассоциировании пользователя со стереотипом.

Можно сформулировать три этапа, которые нужно выполнить при разработке стереотипов:

- определение подгрупп пользователей: необходимо выделить внутри группы пользователей подгруппы, члены которых имеют некоторые однородные характеристики;
- идентификация ключевых характеристик: требуется определить небольшое число ключевых характеристик, позволяющих идентифицировать членов подгруппы пользователей (присутствие или отсутствие этих характеристик должно быть распознаваемо системой);
- представление в виде (иерархически упорядоченных) стереотипов: требуемые характеристики определенных групп пользователей должны быть формализованы в подходящей системе представления. Совокупность представленных характеристик подгруппы поль-

зователей называется «стереотипом» этой подгруппы. Если содержимое одного стереотипа образует подмножество содержимого другого, может быть построена иерархия стереотипов, в которой содержимое стереотипа более высокого уровня наследуется стереотипом более низкого уровня и таким образом представлено один раз.

При использовании стереотипного моделирования может возникнуть следующая проблема: стереотипы могут быть так специализированы, что будут состоять только из одного пользователя, или пользователь вообще не сможет быть классифицирован.

Можно определить четыре модели согласно типу организации связи между стереотипами: луковая (многоуровневая) модель, латуковая модель, многоядерная латуковая модель, ориентированный ациклический граф.

- *«Многоуровневая» или «луковая модель» (onion model)* — иерархическая модель, в которой содержимое стереотипов упорядочено в отношении линейных подмножеств. Пользователи, принадлежащие к стереотипу S, наследуют знание, ассоциированное с более общими стереотипами, чем S (наследование без исключений). В качестве примера можно привести классификацию пользователей, в которой «новички» знают подмножество концептов A; у «продвинутых» пользователей есть знания новичков, но они также знают концепты из подмножества B; наконец, «эксперты» знают концепты из A, B и C.
- *«Латуковая модель» (lettuce model):* для этой модели характерно наличие стереотипа-ядра, являющегося подмножеством всех других стереотипов, которые могут быть независимы друг от друга. Эта модель часто используется для представления знаний пользователей в ПО подобных командам UNIX: пользователи владеют только относительно небольшим множеством основных команд и рядом функционально связанных команд (например, для управления файловой системой, электронной почтой и т.д.).
- *«Многоядерная латуковая модель» (multikernel lettuce model):* это обобщение латуковой модели, в которой существует несколько ядер, являющихся пересечениями некоторых стереотипов.
- *Ориентированный ациклический граф (DAG):* многоядерная модель расширяется тем, что общности из двух или более ядер также сжимаются в ядро. В результате получается иерархия стереотипов, в которой каждый узел может иметь больше одного узла высшего уровня.

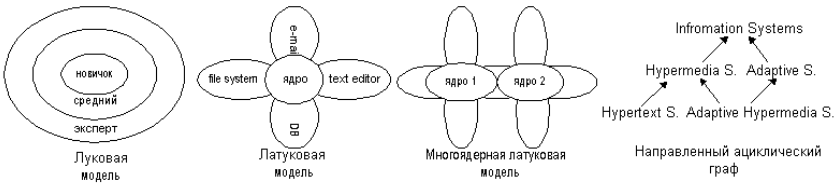


Рис. 4. Модели стереотипов: луковая, латуковая, многоядерная латуковая модель и ориентированный ациклический граф

2.3. Модель адаптации (МА)

Модель адаптации (МА) описывает, как должна происходить адаптация в зависимости от МО и МП. Она состоит из правил адаптации, которые формируют связь между МО и МП и определяют представление генерируемой информации и обновление модели пользователя [11, 12, 15–17].

Обычно правило имеет вид: *если <условие> то <действие>,* где условие может определять происхождение внешнего события, например, обращение к странице, плюс логическое выражение, относящееся к значениям атрибутов из МО или МП. Действием может быть модификация значений атрибутов в МП или присваивание объекту спецификации представления. Также в правиле может указываться «фаза» выполнения, указывающая на момент времени, в который должно применяться правило: до или в течение генерации (фаза “pre”) и после генерации страницы (фаза “post”). Основанием для наличия двух стадий выполнения является то, что сначала может потребоваться осуществить некоторую адаптацию, основываясь на «текущем» состоянии модели пользователя (фаза “pre”), а затем модифицировать модель пользователя после генерации представления страницы (фаза “post”). Также в правиле может быть определено, может ли оно инициировать запуск других правил или нет.

Все правила адаптации подразделяются на «общие» и «специальные» правила. В отличие от общих правил, которые выполняются для всех концептов, в специальных правилах указывается множество концептов, к которым эти правила должны применяться. Специальные правила имеют приоритет над общими правилами, и, таким образом, они используются для определения исключений в общих правилах. В качестве примера можно привести два простых правила, написанных с использованием произвольно выбранного синтаксиса:

Пример 1. Следующее правило определяет, что при обращении к странице для соответствующего концепта в МП устанавливается атрибут «чтение» равным истине в фазе “post”:

<access(C) => C.read:= true; post; true>

Правило также утверждает, что оно запустит другие правила, которые имеют атрибут «чтения» (C.read:= true) в своей левой части.

Пример 2. Следующее правило выражает, что когда пользователь обращается к странице, определяющей концепт, «готовый для чтения», то значение знания для этого концепта становится «изученным» в фазе “pre”:

<(access(C) & C.ready-to-read = true) => C.knowledge-value:=learned; pre; true>

3. ВИРТУАЛЬНЫЙ МУЗЕЙ ИСТОРИИ ИНФОРМАТИКИ В СИБИРИ (СВМ)

В разрабатываемом виртуальном музее истории информатики в Сибири применяются вышеперечисленные методы и технологии АГ. Виртуальный музей создается в виде информационно-поисковой, справочной и обучающей адаптивной гипермедиа-системы, доступной в интернете.

База данных музея поддерживает хранение и обработку информации о следующих объектах: публикациях, документах архива, проектах, ученых-информатиках, коллективах, событиях, конференциях и вычислительной технике. Все вышеперечисленные объекты являются «экспонатами» виртуального музея. Экспонаты, объединенные по тематическому, хронологическому или типологическому критерию, составляют «экспозицию» или «экскурсию». Экспозиции составляют «зал экспозиций», а экскурсии — «зал экскурсий». Оба этих зала являются «открытыми», т.е. доступными для просмотра всеми пользователями музея.

В музее также имеются запасники — залы, доступные только для зарегистрированных пользователей: библиотека, архив, хроника событий, зал ученых-информатиков, зал коллективов, зал проектов, зал вычислительной техники, зал конференций, зал новых поступлений и зал подготовки экспозиций и экскурсий. В библиотеке собраны книги, монографии, сборники статей, учебные и методические пособия, статьи из научных журналов, тезисы конференций и т.д. Архив представляет собой совокупность текстовых, графических, звуковых и видео-материалов. Хроника событий включает описания наиболее выдающихся событий из истории развития информа-

тики в Сибири. Зал информатиков содержит информацию о наиболее выдающихся ученых-информатиках, включая биографии, основные печатные труды и достижения, фото и пр. В зале коллективов содержатся данные о коллективах: группах, лабораториях и институтах. В зале проектов размещены данные о проектах, создаваемых в рамках работ по информатике (темы, системы). В зале вычислительной техники расположены экспонаты, имеющие отношение к вычислительной технике, которая использовалась и разрабатывалась с начала создания Сибирского отделения Академии наук. Зал конференций содержит информацию о различных научных мероприятиях. Новые экспонаты, добавляемые пользователями музея, помещаются в зал новых поступлений. В зале подготовки экспозиций и экскурсий размещаются экспозиции и экскурсии, создаваемые пользователями музея.

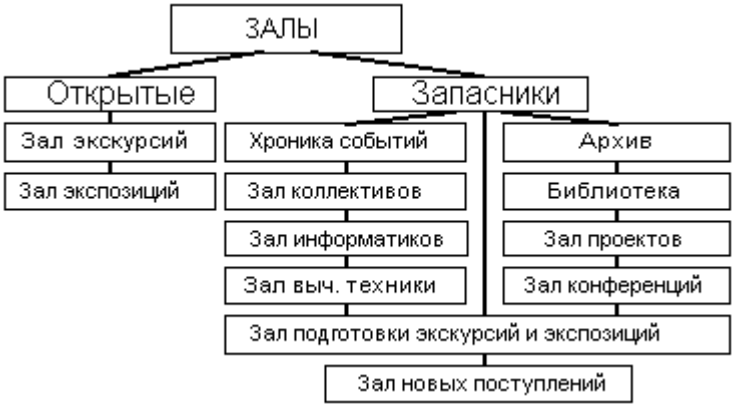


Рис. 5. Структура виртуального музея

Интерфейс музея разрабатывается с учетом его использования различными категориями пользователей. Все пользователи музея подразделяются на две основные категории: незарегистрированные пользователи («посетители») и зарегистрированные пользователи («специалисты»), различающиеся по уровню доступа к информационным ресурсам. «Посетители» могут просматривать только «открытые» залы (залы экскурсий и экспозиций), а «специалисты» — все залы, включая «запасники». «Специалисты» делятся на две группы: группу «простых специалистов» («волонтеры», «экскурсоводы» и «экспозиторы»), работающих только в зале новых поступлений, и группу «музейных работников» («библиотекари», «архивариусы», «хронологи» и т.д.), имеющих права на ввод и редактирование соответствующих

ресурсов музея, во главе этой группы находится главный администратор музея.

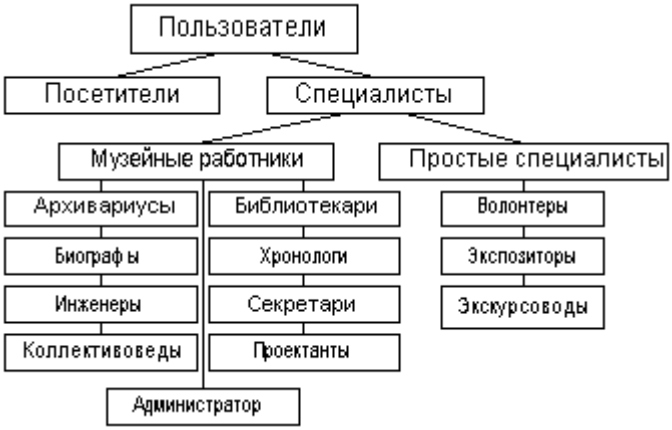


Рис. 6. Категории пользователей музея

К настоящему времени спроектирован и реализован гипермедиа-интерфейс для информационного наполнения музея: просмотра, поиска, ввода и редактирования данных обо всех вышеперечисленных объектах, а также механизм для их связывания. Реализован интерфейс для регистрации и аутентификации пользователей музея и ведения электронной конференции пользователей.

Для адаптивного представления информации в виртуальном музее предполагается использовать такие методы адаптивного представления информации, как дополнительные, предварительные объяснения и сортировка, а также такие методы адаптивной навигационной поддержки, как полное руководство, сортировка ссылок, сокрытие ссылок, аннотирование ссылок и генерирование ссылок.

3.1. Представление знаний предметной области (ПО)

3.1.1. Онтология и модель ПО

Для описания и концептуального структурирования ПО музея был проведен онтологический анализ с целью создания онтологии ПО. Онтологии были первоначально разработаны в области искусственного интеллекта, чтобы облегчить коллективное и многократное использование знаний. Поз-

же, онтологии использовались для интеллектуального поиска знаний в Web как инструмент моделирования семантической информации (метаданных), используемой для аннотирования веб-документов. В данном случае, онтологии используются, чтобы обеспечить максимальную гибкость при представлении знаний ПО. Также они являются основным элементом для достижения разделения представления и содержания, что необходимо для обеспечения адаптивного представления информации.

Основной характерной чертой онтологического анализа является разделение совокупности знаний ПО на классы объектов и определение их онтологий, т.е. совокупности их фундаментальных свойств и отношений. В ходе проведенного онтологического анализа были исследованы объекты, составляющие ПО музея, их характеристики и взаимосвязи. В результате была разработана онтология ПО, или же совокупность словаря терминов, их определений и взаимосвязей между ними. В терминах онтологии в любой системе существуют две основные категории предметов, такие как сами объекты, составляющие систему, и взаимосвязи между этими объектами, характеризующие состояние системы. Понятие взаимосвязи однозначно описывает, или, другими словами, является точным дескриптором зависимости между объектами системы, а термины являются, соответственно, точными дескрипторами самих объектов.

Разработанная онтология состоит из множества классов, представляющих совокупности однородных объектов ПО. Онтологические классы определены для каждого из типов объектов (экспонатов), хранящихся в виртуальном музее. Это, соответственно: классы публикаций, документов архива, проектов, ученых-информатиков, коллективов, событий, конференций, вычислительной техники, экскурсий и экспозиций. Экземпляры вышеперечисленных классов представляют знания об экспонатах музея соответствующих типов.

Для каждого из вышеперечисленных классов определен набор атрибутов (характеристик) и отношений (взаимосвязей), характерный для данного класса объектов. Структуру экземпляров каждого класса представляет соответствующий протофрейм, имеющий следующий вид:

(Имя фрейма:
Имя слота 1 (значение слота 1)
Имя слота 2 (значение слота 2)
. . . . . .
Имя слота К (значение слота К)).

Слоты соответствуют атрибутам и отношениям, определенным для данного класса. При конкретизации фрейма ему и слотам присваиваются кон-

кретные имена, и происходит заполнение слотов. Таким образом, из протофреймов получаются фреймы-экземпляры (экземпляры класса).

Каждый класс (экземпляр класса) имеет следующие основные атрибуты: уникальный универсальный идентификатор (uuid), название (имя), дату, краткое и полное описание, а также служебные данные (пользователь, добавивший объект, дата добавления, права модификации и т.д.). Кроме основных атрибутов, классы могут иметь дополнительные атрибуты и отношения, характерные для объектов данного класса. Так, например, класс «ученых-информатиков» имеет такие дополнительные атрибуты, как биография, фотография, и такие отношения, как список публикаций, список проектов и т.д.

Например, структура экземпляров класса «ученых-информатиков», записанная в форме протофрейма, будет иметь следующий вид:

(Ученые-информатики:
Идентификатор (значение слота 1)
ФИО (значение слота 2)
Дата рождения (значение слота 3)
Образование (значение слота 4)
Ученые степени (значение слота 5)
Ученые звания (значение слота 6)
Кандидатская диссертация (значение слота 7)
Докторская диссертация (значение слота 8)
Научные интересы (значение слота 9)
Текст биографии (значение слота 10)
Фотография (значение слота 11)
Адрес (значение слота 12)
E-mail адрес (значение слота 13)).

Если в качестве значений слотов использовать конкретные данные, то получится фрейм-экземпляр.

Для каждого онтологического класса определена одноименная таблица в базе данных (БД) музея, структура которой соответствует определенному протофрейму (поля таблицы — слоты протофрейма). Информация об экземплярах класса содержится в соответствующей таблице БД, в которой каждому экземпляру соответствует запись в таблице.

После определения онтологии модель ПО строится посредством создания фреймовой сети концептов и межконцептных отношений с использованием классов и отношений, определенных в онтологии. Концепты и межконцептные отношения ПО формируются путем создания фреймов-экземпляров, т.е. заданием значений слотов в протофреймах, и связывания

их друг с другом, используя концептуальный словарь, определенный онтологией. Таким образом, модель ПО является моделью третьего уровня (фреймовой моделью), которая состоит из взаимосвязанных концептов, имеющих внутреннюю структуру (представленную в виде фрейма).

Концепты ПО используются также для поддержания оверлейной модели знаний пользователя. Эта модель динамически обновляется, отражая прогресс в изменяющемся знании пользователя.

3.1.2. Модель представления

Разделение представления и содержания достигается путем задания модели представления: для каждого класса онтологии определены т.н. *шаблоны представления*. Шаблоны определяют, какие части (атрибуты и отношения) экземпляра конкретного класса должны быть включены в представление и в каком порядке, а также их визуальное представление и расположение. Шаблоны дополняются так называемыми *правилами представления*, которые отвечают за генерирование адаптивных конструкций представления.

Шаблоны представляют собой html-код, расширенный вставками PHP-инструкций, позволяющими вставлять в html-код различные управляющие операторы и обращаться к концептам МО (атрибутам и отношениям), которые требуется представить. Веб-представление для определенного концепта (экземпляра класса) динамически генерируется на основе шаблона представления для данного класса и информации о данном концепте, представленной в модели ПО. С помощью вставок PHP-инструкций в html-код во время генерации представления значения требуемых атрибутов и отношений автоматически извлекаются из БД и динамически подставляются в соответствующие места шаблона представления.

Для каждого класса объектов определены два типа шаблонов представления: шаблон для ввода и редактирования и шаблон для просмотра информации об объекте. Шаблон для ввода и редактирования представляет собой набор форм с полями для ввода в БД значений атрибутов и отношений. Шаблон для просмотра является непосредственно шаблоном представления, на его основе генерируется веб-представление экземпляров соответствующего класса. Оба типа шаблонов представляют информацию, разделенную на три части: общая, дополнительная и служебная. Общую информацию составляют такие атрибуты, как название (имя) объекта, дата, краткое описание и т.д. Дополнительная информация включает в себя, например, полное описание. Служебная информация содержит данные о пользователе, добавившем объект в БД, дату добавления, права изменения и т.д.

Что касается «полного описания» объекта, то оно тоже является своего рода шаблоном, но не предопределенным в системе, а создаваемым самим пользователем для данного объекта. Шаблон создается пользователем в режиме on-line с помощью разработанного специального html-редактора. Редактор поддерживает стандартные функции форматирования текста, включения графической, аудио- и видео-информации. Имеется возможность предварительного просмотра текста, генерируемого по создаваемому шаблону, в процессе ввода и редактирования. Реализована функция автоматического и динамического включения информации (атрибутов и отношений) об объектах МО в представление. Реализована возможность выбора для каждого типа объектов соответствующих атрибутов, а также возможность указания способа подстановки (простая подстановка или гиперссылка).

Таким же образом создаются экспозиции, с тем лишь отличием, что их веб-представление имеет структуру, состоящую из набора связанных гипермедиа-страниц. Шаблоны для этих гипермедиа-страниц конструируются пользователем аналогичным образом с помощью того же html-редактора.

Таким образом, в системе используется третий метод (прямая связь) для организации связи между концептами модели ПО и гиперпространством системы (гипермедиа-страницами), т.е. гиперпространство строится непосредственно исходя из структуры ПО. Каждый концепт модели ПО представлен динамической гипермедиа-страницей или гипердокументом, которые генерируются налету, исходя из внутренней структуры концепта.

3.2. Моделирование пользователя

Для зарегистрированных пользователей поддерживается МП, состоящая из модели индивидуальных сведений, модели категорий, модели предпочтений и модели знаний.

Статическая *модель индивидуальных сведений* включает информацию об идентификационных данных пользователя (идентификатор и пароль для входа в систему), ФИО, почтовом и e-mail адресах. Эта информация хранится в соответствующей таблице БД и извлекается во время авторизации пользователя.

Модель категорий моделирует права доступа пользователя к информационным ресурсам музея. Она реализуется с помощью статической стереотипной модели: для каждой категории пользователей музея определен одноименный стереотип, характеризующийся определенными значениями атрибутов. В качестве атрибутов используются имена типов ресурсов БД

(публикации, проекты, события и т. д.), значениями атрибутов являются права для доступа к соответствующему типу ресурсов (просмотр, добавление, изменение, удаление и их комбинации). В таблице приведен набор стереотипов пользователей с указанием прав доступа к различным типам ресурсов, где просмотр обозначен как «П», добавление — «Д», изменение — «И» и удаление — «У».

	Экскурсии	Экспозиции	Публикации	Док-ты архива	Проекты	Информатики	Коллективы	События	Конференции	Выч. техника
Посетители	П	П								
Специалисты	П	П	П	П	П	П	П	П	П	П
Волонтеры	П	П	ПД	ПД	ПД	ПД	ПД	ПД	ПД	ПД
Экскурсоводы	ПДИ	П	П	П	П	П	П	П	П	П
Экспозиторы	П	ПДИ	П	П	П	П	П	П	П	П
Библиотекари	П	П	ПДИ	П	П	П	П	П	П	П
Архивариусы	П	П	П	ПДИ	П	П	П	П	П	П
Проектанты	П	П	П	П	ПДИ	П	П	П	П	П
Биографы	П	П	П	П	П	ПДИ	П	П	П	П
Коллективоведы	П	П	П	П	П	П	ПДИ	П	П	П
Хронологи	П	П	П	П	П	П	П	ПДИ	П	П
Секретари	П	П	П	П	П	П	П	П	ПДИ	П
Инженеры	П	П	П	П	П	П	П	П	П	ПДИ
Администраторы	ПДИУ	ПДИУ	ПДИУ	ПДИУ	ПДИУ	ПДИУ	ПДИУ	ПДИУ	ПДИУ	ПДИУ

При входе в систему каждый пользователь идентифицируется как принадлежащий к определенному стереотипу (категории), исходя из структуры его идентификатора, и при помощи модели категорий получает соответствующие права доступа к информационным ресурсам.

Модель предпочтений моделирует различные предпочтения пользователей, в частности, способ представления информации (использование только текста, графики, звука, видео и т.д.). Она реализуется с помощью статической стереотипной модели, атрибутами которой являются вышеперечисленные способы представления информации, а значениями — истина или ложь.

Модель знаний используется для моделирования знаний ПО пользователя и реализуется посредством оверлейной модели. Система получает данные для модели, отслеживая историю навигации пользователя. Знание пользователя представляется как подмножество знания эксперта ПО: для каждого концепта МО определен набор пар атрибут–значение, где в качестве ат-

рибутов используются атрибут "чтение" (прочитан, не прочитан) и атрибут "знание" концепта (изучен, не изучен). Модель знаний для каждого пользователя представляется в виде следующего вектора, содержащего информацию о множестве прочитанных и изученных концептов ПО:

(users_uuid, concept1, read-value1, known-value1,… conceptN, read-valueN, known-valueN),

где users_uuid — идентификатор пользователя, concept — идентификатор концепта, read-value — значение чтения, known-value — значение знания. По мере просмотра пользователем информации модель автоматически обновляется.

ЗАКЛЮЧЕНИЕ

В статье представлен проект создания виртуального музея истории информатики в Сибири, разрабатываемого в виде информационно-поисковой, справочной адаптивной гипермедиа-системы, доступной в интернете. Работа над данным проектом ведется коллективом сотрудников ИСИ СО РАН, ИМ СО РАН и НГУ при финансовой поддержке РГНФ (02-05-12010).

В статье представлены методы и технологии АГ, описаны архитектура АГС и ее основные компоненты (модель предметной области, модель пользователя и модель адаптации), рассмотрены вопросы моделирования предметной области и моделирования пользователя. Описанные методы и технологии адаптивной гипермедиа, моделирования предметной области и моделирования пользователя были использованы при создании данного виртуального музея. В статье обсуждаются вопросы представления знаний ПО, построения онтологии и модели ПО, а также разработки модели пользователя.

СПИСОК ЛИТЕРАТУРЫ

1. **Волянская Т.А.** Виртуальный музей истории информатики в Сибири // Материалы Междунар. конф. молодых ученых по математическому моделированию и информационным технологиям. — Новосибирск, 2002. — С. 49.
2. **Волянская Т.А.** Методы адаптации гипермедиа и их применение при создании виртуального музея истории информатики в Сибири // Материалы XL Междунар. науч. студенческой конф. «Студент и научно-технический прогресс». — Новосибирск, 2002. — С. 173–174.

3. **Волянская Т.А.** Методы и технологии адаптивной гипермедиа // Современные проблемы конструирования программ. — Новосибирск, 2002. — С. 38–68 .
4. **Касьянов В.Н., Несговорова Г.П., Волянская Т.А.** Виртуальный музей истории информатики в Сибири // Электронные изображения и виртуальные искусства. — Киев, 2002. — С. 242–250.
5. **Касьянов В.Н., Несговорова Г.П., Волянская Т.А.** Виртуальный музей истории информатики в Сибири // Современные проблемы конструирования программ. — Новосибирск, 2002. — С. 169–181.
6. **Brusilovsky P.** Adaptive Hypermedia // User Modeling and User-Adapted Interaction. — 2001. — Vol 11. –P. 87–110.
7. **Brusilovsky P.** Adaptive hypermedia, an attempt to analyze and generalize // Lect. Notes. Comput. Sci. — 1996. — Vol. 1077. — P. 288–304.
8. **Brusilovsky P.** Methods and techniques of adaptive hypermedia // User Modeling and User-Adapted Interaction. — 1996. — Vol 6. — P. 87–129.
9. **Brusilovsky P., Cooper D. W.** Domain, Task, and User Models for an Adaptive Hypermedia Performance Support System // Proc. of 2002 Internat. Conf. on Intelligent User Interfaces. — San Francisco, CA, 2002. — P. 23–30.
10. **De Bra P.** Adaptive Hypermedia on the Web: Methods, techniques and applications // Proc. of the AACE WebNet'98 Conf. — Orlando, Fl., 1998. — P. 220–225.
11. **De Bra, P., Brusilovsky P., Houben G.-J.** Adaptive Hypermedia: From Systems to Framework // ACM Computing Surveys. — 1999. — Vol. 31, N 4. — Article N 12.
12. **De Bra P., Houben G.J., Wu H.** AHAM: A Dexter-based Reference Model for Adaptive Hypermedia // Proc. of the ACM Conf. on Hypertext and Hypermedia. — Darmstadt, Germany, 1999. — P.147–156.
13. **Kobsa A.** User Modeling: Recent Work, Prospects and Hazards // Adaptive User Interfaces: Principles and Practise. — Amsterdam, North Holland Elsevier. — 1993.
14. **Kobsa A.** User Modeling in Dialog Systems: Potentials and Hazards // AI & Society. — 1990. — Vol. 4. — P. 214–240.
15. **Wu H., De Bra P., Aerts A., Houben G.J.** Adaptation Control in Adaptive Hypermedia Systems // Lect. Notes. Comput. Sci. — 2000. — Vol. 1892. — P. 250–259.
16. **Wu H., De Kort E., De Bra P.** Design Isuuses for General-Purpose Adaptive Hypermedia Systems // Proc. of the ACM Conf. on Hypertext and Hypermedia. — Aarhus, Denmark, 2001. — P. 141–150.
17. **Wu H., Houben G.J., De Bra P.** Supporting User Adaptation in Adaptive Hypermedia Applications // On-line Conf. and Informatiewetenschap 2000. — De Doelen, Rotterdam, 2000.

А.А. Дунаев, А.Э. Кель, И.В. Лобив, Ф.А. Мурзин,
О.Н. Половинко, Е.С. Черемушкин

ВИЗУАЛИЗАЦИЯ ГЕНЕТИЧЕСКОЙ ИНФОРМАЦИИ*

ВВЕДЕНИЕ

В настоящее время проведены основные экспериментальные работы по секвенированию нуклеотидных последовательностей. Для хранения получаемой первичной информации созданы и постоянно пополняются такие специализированные банки данных, как EMBL и GenBank. В то же время, несмотря на наличие большого количества отсеквенированных последовательностей, наши представления о принципах их организации весьма ограничены. Поэтому одним из ведущих направлений молекулярной биологии в последнее время становится компьютерный анализ генетических текстов [1,2].

Проблематика идентификации структурно-функциональной организации генома наряду с такими вопросами, как распознавание интронов, экзонов или сайтов сплайсинга, включает в себя и круг задач, связанных с регуляцией транскрипции генов позвоночных [3].

Ввиду больших объемов генетических текстов возникает необходимость в визуализации генетической информации. Визуализация генетических текстов может стать необходимым шагом в процессе решения различных генетических задач, например задач распознавания специфичных участков ДНК (генов, сайтов и т.д.) [4]. Визуальный анализ биологических последовательностей [5, 6, 7] дает возможность определить структуру информации, закодированной в геноме, а также корректно выбрать метод для анализа этой структуры.

1. АЛГОРИТМЫ ВИЗУАЛИЗАЦИИ

Авторами были разработаны несколько алгоритмов для представления генетических текстов в графической форме и пакет программ, реализующий

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

данные возможности. Ниже описаны некоторые из реализованных алгоритмов.

1.1. Визуализация частот нуклеотидов

Пусть $\overline{S}$ — последовательность в четырехбуквенном алфавите A, C, G, T. Обозначим k-й элемент последовательности как s_k, и длину последовательности — M.

Пусть дано N. Обозначим через $BL_N[i]$ подпоследовательность $\overline{S}$ длины N, начинающуюся с i-й позиции, т.е. $BL_N[i] = s_i \ldots \ s_{N+i-1}$.

Пусть $n_A[i,N]$, $n_C[i,N]$, $n_G[i,N]$, $n_T[i,N]$ — количества букв A, C, G, T рассматриваемой подпоследовательности $BL_N[i]$ соответственно. Если i, N заранее известны, то будем писать для краткости n_A, n_C, n_G, n_T.

Легко видеть, что $n_T = N - (n_A + n_C + n_G)$. Это означает, что достаточно изучать поведение трех компонентов. Отсюда могут быть вычислены частоты $p_A = n_A / N$, $p_C = n_C / N$, $p_G = n_G / N$.

Введем $\overline{p}_A = f(p_A)$, $\overline{p}_C = f(p_C)$, $\overline{p}_G = f(p_G)$, где $f(x) = \mathrm{int}(255 \times x)$. Тогда тройка $\langle \overline{p}_A,\ \overline{p}_C,\ \overline{p}_G \rangle$ может быть рассмотрена как вектор компонентов цвета $\langle R,\ G,\ B \rangle$ соответственно.

Цветное изображение может быть задано тремя матрицами $S = (S_R, S_G, S_B)$, $S_R = S_R(i,j)$, $S_G = S_G(i,j)$ $S_B = S_B(i,j)$, $0 \le i \le n-1$, $0 \le j \le m-1$. Обычно значения $S_R(i,j), S_G(i,j), S_B(i,j)$ лежат в диапазоне от 0 до 255. Набор троек $\{ (r,g,b) : 0 \le r,g,b < 255 \}$ называется цветовым кубом. Наша задача состоит в построении изображения, отражающего адекватность данных частот.

Предположим, что даны две позиции i_1, i_2 на последовательности $\overline{S}$, $i_1 - i_2 \le n \cdot m$ и $i_1 \le k \le i_2$. Далее, рассматриваемое окно $BL_N[k]$ движется вдоль данной последовательности $\overline{S}$. Затем мы получаем соответствующую тройку $\langle \overline{p}_A,\ \overline{p}_C,\ \overline{p}_G \rangle$ для каждой позиции k.

Поэтому запишем

$$\langle \overline{p}_A,\ \overline{p}_C,\ \overline{p}_G \rangle = \langle \overline{p}_A(k),\ \overline{p}_C(k),\ \overline{p}_G(k) \rangle = \langle R(k),\ G(k),\ B(k) \rangle .$$

Теперь мы можем создать следующее изображение

$$S_R(i,j)=\begin{cases} R(i_1+m\cdot i+j-1),\ i_1+m\cdot i+j-1\le n\cdot m;\\ 0,\ i_1+m\cdot i+j-1>n\cdot m;\end{cases}$$

$$S_G(i,j)=\begin{cases} G(i_1+m\cdot i+j-1),\ i_1+m\cdot i+j-1\le n\cdot m;\\ 0,\ i_1+m\cdot i+j-1>n\cdot m;\end{cases}$$

$$S_B(i,j)=\begin{cases} B(i_1+m\cdot i+j-1),\ i_1+m\cdot i+j-1\le n\cdot m;\\ 0,\ i_1+m\cdot i+j-1>n\cdot m.\end{cases}$$

Осуществляется последовательное заполнение изображения пикселями в процессе обозрения компонентов $\langle R,\ G,\ B\rangle$.

Вначале мы заполняем верхний ряд, т.е. $i=0$, затем первый и т.д. Аналогично, двигаясь вдоль последовательности $\overline{S}$, мы можем получить второе изображение, третье и т.д. Как результат, получаем последовательность изображений, которые образуют видеоряд и могут быть представлены в виде AVI-файла.

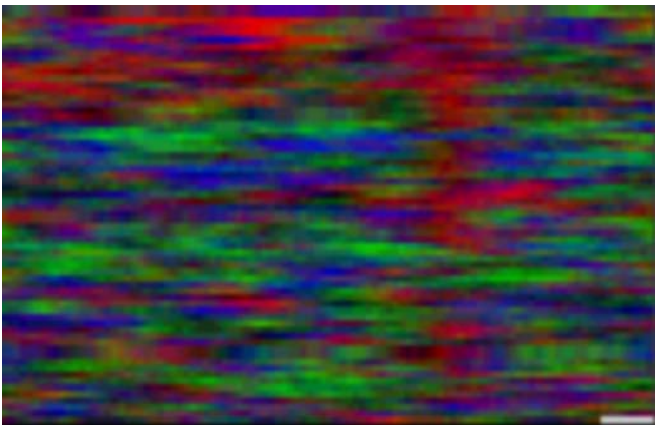


Рис. 1. Последовательное заполнение изображения в процессе обозрения компонентов $\langle R,\ G,\ B\rangle$

Этот процесс напоминает действие сглаживающего одномерного фильтра на последовательность, а потом вывод по строкам. На рисунке видна нерегулярная структура ДНК, но, тем не менее, прослеживаются некоторые закономерности.

Регулируя параметры алгоритма (например, ширину изображения или длину окна), можно увидеть некоторые общие участки. Неплохо прослеживается неоднородность GC-состава ДНК. Например, в центре больше зеленого и синего, что указывает на известный факт, что кодирующие области более GC-богаты, чем некодирующие.

1.2. Визуализация структуры нуклеотидной последовательности

Пусть дана функция $g : \{A,C,G,T\} \to \{i : 0 \le i \le 255\}$. Тогда наша последовательность $\overline{S}$ создает последовательность целых чисел по следующему правилу

$$g\left[\overline{S}\right] = g(s_1)g(s_2)g(s_3)\ldots$$

Каждые 3 числа, стоящие рядом, могут рассматриваться, как компоненты цвета, т.е. мы имеем следующую последовательность троек

$$\langle g(s_1)g(s_2)g(s_3)\rangle, \langle g(s_4)g(s_5)g(s_6)\rangle, \langle g(s_7)g(s_8)g(s_9)\rangle\ldots =$$
$$= \langle R_1, G_1, B_1\rangle,\ \langle R_2, G_2, B_2\rangle,\ \langle R_3, G_3, B_3\rangle,\ldots$$

Аналогично, двигаясь вдоль последовательности, получаем последовательность изображений.

Также можно рассмотреть другие функции

$$g_2 : \{A,C,G,T\}^2 \to \{i : 0 \le i \le 255\} \text{ или } g_3 : \{A,C,G,T\}^3 \to \{i : 0 \le i \le 255\}.$$

В этом случае рассматриваются пары и тройки на последовательностях. Как известно, они более информативны.

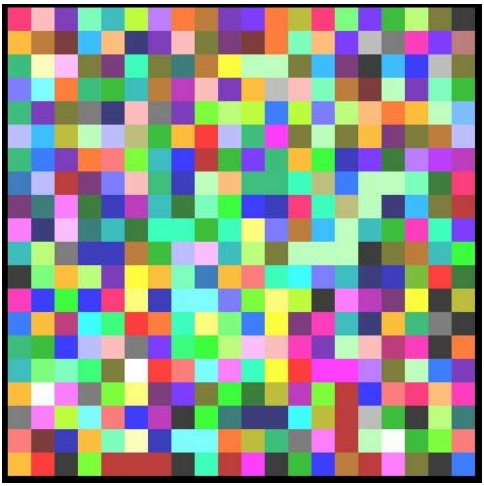


Рис. 2. Визуализация функции $g : \{A, C, G, T\} \to \{i : 0 \le i \le 255\}$

Отчетливо видна нерегулярная структура ДНК. По большей части генетическая информация похожа на высокочастотный шум, так что для анализа такой информации целесообразно использовать соответствующие методы высокочастотного анализа в совокупности с методами, опирающимися на реальные экспериментальные данные.

1.3. Визуализация в трехмерном пространстве

Рассмотрим последовательность троек, описанную в первом алгоритме, $\langle \overline{p}_A(k),\ \overline{p}_C(k),\ \overline{p}_G(k) \rangle,\ k \ge 1$. Они могут быть представлены координатами в трехмерном пространстве.

Предположим, что дана функция $h : [0,1]^3 \to \{i : 0 \le i \le 255\}^3$. Понятно, что она может быть представлена в виде

$$h(x, y, z) = \langle h_R(x, y, z),\ h_G(x, y, z),\ h_B(x, y, z) \rangle .$$

В итоге получаем трехмерное изображение, которое позволяет лучше увидеть структуру последовательности $\overline{S}$. Были использованы различные формы функции визуализации.

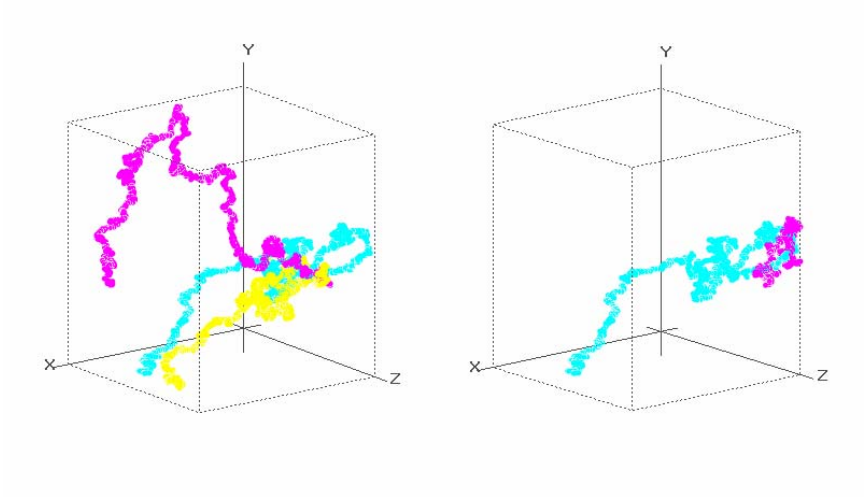


Рис. 3. Визуализация промоторов с помощью функции $h : [0,1]^3 \to \{i : 0 \le i \le 255\}^3$

Слева изображены три промотора для одного и того же гена *c-myc* у разных организмов: человека, мыши и крысы. Видна схожесть в поведении этих трех кривых. Справа — промоторы разных генов *a` actin* и *c-myc* у человека. Видно, что поведение кривых различается. В данном случае координатами точек являются наши p_A, p_C, p_G. Кубы на рисунках — единичные.

2. ВИЗУАЛЬНЫЙ АНАЛИЗ ВЫБОРКИ ПРОМОТОРОВ И РАСПОЗНАВАНИЕ ССТФ

Далее можно проанализировать промоторы [2]. В силу зависимости между сайтами связывания транскрипционных факторов (ССТФ), относящихся к похожим транскрипционным факторам, мы использовали специальную формулу для вычисления степени похожести промоторов, затем отсортировали их и выявили наиболее статичные участки.

Мы взяли выборку промоторов генов, специфично экспрессирующихся в печени ($\overline{S}^1 \dots \overline{S}^K$) одинаковой длины L = 100 и рассчитали попарную похожесть при помощи нашей метрики. Относительное положение промоторов мы вычисляли, пользуясь известным положением старта транскрипции.

Потом находим путь ($p_1..p_{K-1}$) со свойством $\sum_{i=1}^{K-1} sim\left(\overline{S}^{p_i}, \overline{S}^{p_{i+1}}, L\right) \to \max$, где sim — похожесть между p_i и p_{i+1}.

Применяя правило ближайшего соседа, получаем приближенное решение. Потом на всех парах последовательностей $\overline{S}^{p_i}, \overline{S}^{p_{i+1}}$ мы ищем T непересекающихся фрагментов ($B_1 \ldots B_T$) длины $P < L$ с максимальной похожестью $sim^*(i,j) = sim\left(\overline{S}^{p_i}, \overline{S}^{p_{i+1}}, j, P\right)$, где i — номер последовательности в полученном пути ($p_1..p_{K-1}$) и j — старт фрагмента. На следующем рисунке показана визуализация этих данных.

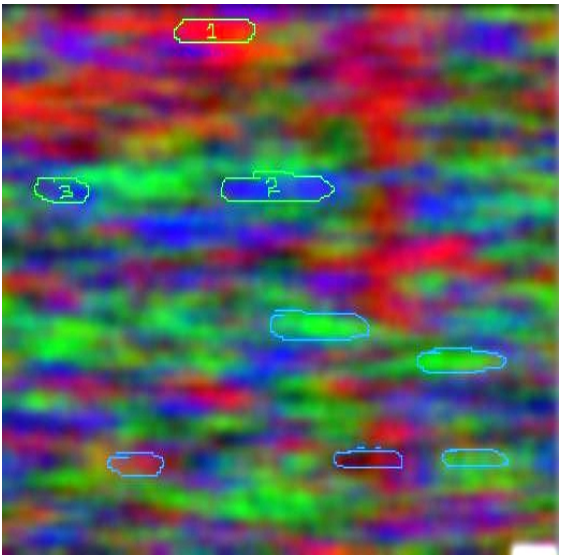


Рис. 4. Визуализация полученных результатов

Взята выборка промоторов генов, специфично экспрессирующихся в печени. После их сортировки с целью минимизации описанного выше функционала мы визуализировали их, используя первый алгоритм визуализации. Обведенные участки соответствуют высокогомологичным областям.

3. КРАТНОМАСШТАБНЫЙ АНАЛИЗ

Кратномасштабный анализ представляет собой широко известный математический метод, базирующийся на применении вейвлет-преобразования и позволяющий, в частности, эффективно исследовать одномерные сигналы [8].

В зависимости от конкретного приложения, исходные данные могут быть представлены в различных форматах. С другой стороны, для выполнения преобразования наиболее удобным является формат представления данных, при котором отсчеты записаны последовательно в виде чисел с плавающей запятой в двоичном формате. В таком случае становится возможным выполнять вычисления непосредственно после чтения фрагмента файла.

Теперь рассмотрим подготовку к обработке нуклеотидной последовательности. Нуклеотидная последовательность является, по сути, словом, состоящим из букв «генетического алфавита» — нуклеотидов A, C, T и G. Очевидно, такое представление малопригодно для какого-либо численного анализа, поэтому выполняется преобразование последовательности нуклеотидов к одномерному массиву чисел. В простейшем случае каждой букве алфавита сопоставляют число (иногда буквы группируют по две или по три, такой метод называется методом простого сопоставления). Полученная последовательность чисел уже может быть рассмотрена в качестве исходных данных для применения численных методов. Преобразованные данные записываются в файл в естественном формате, который после исчерпания данных в исходном файле дополняется нулями до оптимального размера, зависящего от конкретного вейвлета, применяемого в данный момент.

Для анализа данных используется видоизмененное быстрое вейвлет-преобразование, опирающееся на метод кратномасштабного анализа, разработанного Малла и Мейером, известного также, как пирамидальный алгоритм Малла. Были реализованы несколько вычислительных модулей, представляющих различные классы вейвлетов (вейвлеты Добеши: DB4, DB6, DB8; вейвлеты Хаара).

Результат вычислений — несколько векторов, являющихся приближениями одного и того же исходного вектора. Применительно к исследованиям нуклеотидных цепочек существуют несколько методов визуализации информации подобного рода. В настоящей работе был выбран наиболее наглядный, с точки зрения авторов, способ, который заключается в следующем (ниже показано главное окно программы).

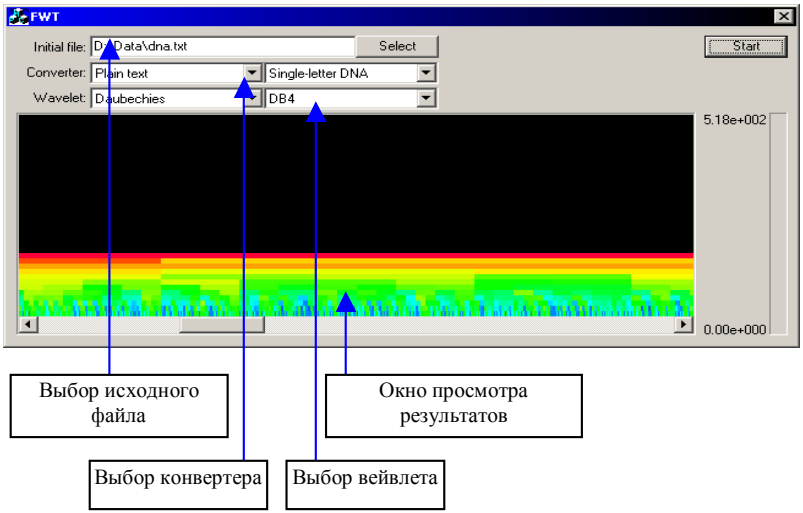


Рис. 5. Результат применения вейвлет-преобразования DB4

Среди всех значений, содержащихся в полученных массивах, выбирается минимальное и максимальное значения. После этого строится цветовая шкала соответствия значения оттенку цвета H в системе цветовых координат HSV. Минимальному значению соответствует цвет с оттенком 0, максимальному — с оттенком 360. После этого массивы отображаются на плоскости рядами цветных точек; цвет точки соответствует значению элемента массива. Такой способ отображения дает возможность визуально выделять характерные участки в массиве данных.

Реализованная программа позволяет работать с файлами объемом до 300 Мбайт. Проведенные предварительные исследования показали, что визуализация результатов вейвлет-преобразования, примененного к сигналам, ассоциированным с генетической последовательностью, может оказаться значительно более информативным методом визуализации, по сравнению с рассмотренными ранее.

СПИСОК ЛИТЕРАТУРЫ

1. **Doolittle R. F.** Microbial genomes opened up // Nature. — 1997. — Vol. 392. — P. 339–342.
2. **Maley L. E., Marshall C. R.** The coming of age of molecular systematics // Science. — 1998. — Vol. 279. — P. 505–506.
3. **Ulyanov A., Stormo G.** Multi-alphabet consensus algorithm for identification of low specificity protein-DNA interactions // Nucleic Acids Res. — 1995. — Vol. 23. — P. 1434–1440.
4. **Kel A.E., Kondrakhin Y.V., Kolpakov Ph.A., Kel O.V., Romashenko A.G., Wingender E., Milanesi L., Kolchanov N.A.** Computer tool FUNSITE for analysis of eukaryotic regulatory genomic sequences // Proc. Third Internat. Conf. Intelligent Systems Molec. Biol. — 1995. — P.197–205.
5. **Jeffrey H. J.** Chaos game representation of gene structure // Nucleic Acids Res. — 1990. — Vol. 18. — P. 2163–2170.
6. **Burma P. K., Raj A., Deb J.K., Brahmachari S. K.** Genome analysis: a new approach for visualization of sequence organization in genomes // J. Biosci.. — 1992. — Vol. 17. — P. 395–411.
7. **Solovyev V. V.** Fractal graphical representation and analysis of DNA and protein sequences // Biosystems. — 1993. — Vol. 30. — P. 137–160.
8. **Астафьева Н. М.** Вейвлет-анализ: основы теории и примеры применения // Успехи физических наук. — 1998. — Т. 166, № 11. — С. 1145–1170.

М. А. Иванов

ПРИМЕНЕНИЕ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЙ В КОДИРОВАНИИ ИЗОБРАЖЕНИЙ*

ВВЕДЕНИЕ

Первое упоминание о вейвлетах появилось в литературе по цифровой обработке и анализу сейсмических сигналов в работах А. Гроссмана и Ж. Морлета. Однако набор базисных функций был избыточным, так как основной интерес авторов был направлен на анализ сигналов. Далее, математик И. Мейер показал существование вейвлетов, образующих ортонормальный базис в $L^2(R)$. Дискретизация вейвлет-преобразования была описана в статье И. Добеши, которая объединила разработки математиков и специалистов в области обработки сигналов. Добеши разработала семейство вейвлет-преобразований, имеющих максимальную гладкость для данной длины фильтра. Популярность вейвлетов увеличилась после введения С. Маллатом концепции кратномасштабного анализа. Он же, по-видимому, первым применил вейвлеты для кодирования изображений. И И. Добеши, и С. Маллат показали, что практическое выполнение вейвлет-преобразований осуществляется посредством двухполосного банка фильтров анализа-синтеза известного ранее в теории субполосного кодирования. Эта теория может быть описана в терминах вейвлетов. Главное различие между этими двумя направлениями заключается в критериях построения фильтров.

Некоторые идеи теории вейвлетов частично были разработаны уже очень давно. Например, А. Хаар опубликовал в 1910 году полную ортонормальную систему базисных функций с локальной областью определения. Эти функции называются теперь вейвлетами Хаара.

В настоящее время исследования в области вейвлетов ведутся по многим направлениям. В частности, разработана лифтинговая схема выполнения вейвлет-преобразований, имеющая ряд преимуществ по сравнению с традиционной. Активно исследуется целочисленное вейвлет-преобразование.

Цель настоящей статьи — изложить основы теории вейвлет-преобразований и дать обзор основных способов применения вейвлет-преобразований к задаче кодирования изображений.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

Несмотря на то, что теория вейвлет-преобразований уже в основном разработана, точного определения, что же такое «вейвлет», какие функции можно назвать вейвлетами, насколько известно, не существует. Обычно под вейвлетами понимаются функции, сдвиги и растяжения которых образуют базис многих важных пространств, в том числе и $L^2(R)$. Эти функции являются компактными как во временной, так и в частотной области. Вейвлеты непосредственно связаны с кратномасштабным анализом сигналов. Вейвлеты могут быть ортогональными, полуортогональными, биортогональными. Эти функции могут быть симметричными, асимметричными и несимметричными. Различают вейвлеты: с компактной областью определения и не имеющие таковой. Некоторые функции имеют аналитическое выражение, другие — быстрый алгоритм вычисления связанного с ними вейвлет-преобразования. Вейвлеты различаются также степенью гладкости. Для практики желательно было бы иметь ортогональные симметричные или асимметричные вейвлеты. К сожалению, доказана теорема о том, что такими вейвлетами являются лишь вейвлеты Хаара. Функции Хаара не обладают достаточной гладкостью и не подходят для большинства приложений, поэтому для кодирования изображений обычно используют биортогональные вейвлеты.

Может показаться, что вейвлеты не являются чем-то фундаментально новым. В самом деле, сходные идеи появлялись на протяжении последних десятилетий: субполосное кодирование, успешно применяемое при кодировании речи, пирамидальные схемы кодирования изображений, преобразование и функции Габора (вейвлеты Габора). С развитием теории вейвлетов произошло как бы объединение и взаимообогащение этих идей, что привело к качественно новому результату.

ТЕОРИЯ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЙ

Одним из основных средств обработки сигналов является линейное преобразование. Традиционно, кодеры, основанные на линейном преобразовании, делят на кодеры с преобразованием и субполосные.

Кодирование с линейным преобразованием основывается на применении ортогонального линейного преобразования. Классическим примером такого преобразования является дискретное преобразование Фурье (ДПФ), которое разлагает сигнал на синусоидальные компоненты. Другими примерами являются дискретное косинусное преобразование (ДКП) и преобразование Корунена—Лоэва (ПКЛ). Эти преобразования находятся путем вы-

числения свертки сигнала конечной длины с семейством базисных функций. В результате получается ряд коэффициентов, которые подвергаются дальнейшей обработке. Обычно эти преобразования применяются к неперекрывающимся блокам исходного сигнала, что приводит к дополнительным искажениям, хотя есть модификации алгоритмов, ликвидирующие данный недостаток.

Субполосное кодирование реализуется посредством свертки исходного сигнала с несколькими полосовыми фильтрами и последующей децимации результата. Совокупность набора фильтров с дециматорами называется банком фильтров. Каждый получившийся в результате сигнал несет в себе информацию об определенной спектральной составляющей исходного сигнала при некотором пространственно-временном масштабе. Для обратного синтеза сигнала выполняются операции интерполяции субполосных сигналов, их фильтрация и сложение. Большинство методов синтеза фильтров направлено на устранение эффекта наложения спектров (aliasing), возникающего при децимации. В пространственной области этот эффект проявляется в виде дискретной структуры восстановленного изображения. Идеальный банк фильтров должен состоять из фильтров с прямоугольной характеристикой, предотвращающих вместе с тем aliasing. Такие фильтры, однако, приводят к так называемому эффекту Гиббса, вследствие которого искажения сильно заметны.

Сформулируем некоторые важные требования, предъявляемые к преобразованиям, используемым при кодировании изображений.

1. Масштаб и ориентация

В изображениях имеются объекты различных размеров. Поэтому, преобразование должно позволять анализировать изображение одновременно и независимо на различных масштабах, иначе говоря, допускать кратномасштабный анализ.

Для двухмерного сигнала некоторая спектральная область соответствует определенному масштабу и ориентации. Ориентация базисных функций определяет способность преобразования корректно анализировать ориентированные структуры, типичные для изображений (линии, контуры). Таким образом, для решения задачи анализа желательно иметь преобразование, которое делило бы входной сигнал на локальные частотные области.

2. Пространственная локализация

Необходимость в пространственной локализации преобразования возникает тогда, когда информация о местоположении деталей изображения является важнейшей. Эта локальность, однако, не должна быть абсолютной,

блочной, как при ДКП, так как это ведет к потере свойства локальности в частотной области. В 1946 году Д. Габор предложил класс линейных преобразований, которые обеспечивают локальность и в частотной, и во временной областях. Вейвлеты являются еще одним примером функций, локализованных в пространственной и частотной областях.

3. Ортогональность

Вообще говоря, преобразование не обязательно должно быть ортогональным. Ортогональность функций лишь упрощает вычисления.

4. Быстрые алгоритмы вычисления

Это наиболее важное свойство, предъявляемое к линейным преобразованиям, применяемым для кодирования изображений, так как невозможность применения в реальном времени сводит на нет все их положительные свойства.

Непрерывное вейвлет-преобразование

Основным средством анализа стационарных непрерывных сигналов является преобразование Фурье непрерывного времени (CTFT):

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx ,$$

где $f(x)$ — исходный сигнал, а $F(\omega)$ — его преобразование Фурье.

С практической точки зрения, CTFT имеет ряд недостатков. Во-первых, для получения преобразования по одной частоте требуется информация о сигнале в любой момент времени. Во-вторых, на практике не все сигналы стационарны. Пик в сигнале во временной области распространится на все частоты. Для преодоления этих недостатков вводится оконное преобразование Фурье (STFT):

$$STFT_f(\omega, b) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} w(x-b) dx ,$$

где $w(x-b)$ — локальная функция окна, которая сдвигается вдоль временной оси. Преобразование становится зависимым от времени, и в результате получается пространственно-временное описание сигнала. Недостаток STFT в том, что в нем используется фиксированное окно, которое невозможно адаптировать к локальным особенностям сигнала.

Вейвлет-преобразование, рассмотренное ниже, решает эту проблему. Непрерывное вейвлет-преобразование (CTWT) есть скалярное произведение исходной функции и базисных функций:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi(\frac{x-b}{a}),\ a \in R^+, b \in R\ ,$$

так что

$$CTWT_f(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty}\psi(\frac{x-b}{a})f(x)dx\ .$$

Базисные функции $\psi_{a,b} \in L^2(R)$ определены на некотором интервале. Они и называются вейвлетами. Их можно рассматривать как масштабирование и сдвиг некоторой функции-прототипа $\psi(x)$. Параметр b отвечает за расположение во времени, а a — за масштаб. Большие значения a соответствуют низким частотам, меньшие — высоким.

Для того чтобы было возможно обратное преобразование, функция $\psi(x)$ должна удовлетворять следующему условию:

$$C_\psi = \int_0^\infty \frac{|\Psi(\omega)|^2}{\omega} d\omega < \infty\ ,$$

где $\Psi(\omega)$ — преобразование Фурье функции $\psi(x)$.

Если $\psi(x)$ — локальная функция, удовлетворяющая данному условию, то ее среднее значение равно нулю и обратное преобразование имеет вид:

$$f(x) = \frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_0^\infty CTWT_f(a,b)\frac{1}{\sqrt{a}}\psi(\frac{x-b}{a})\frac{dadb}{a^2}\ .$$

Параметры a и b изменяются непрерывно, поэтому множество базисных функций избыточно. Необходима дискретизация этих значений при условии сохранения возможности обратного преобразования. Нетрудно показать, что при следующей дискретизации это условие соблюдается:

$$a = a_0^m;\ b = nb_0a_0^m,\ m,n \in Z, a_0 > 1, b_0 \neq 0.$$

Возможен произвольный выбор b_0. Выберем $b_0 = 1$. Видно, что параметр местоположения во времени зависит от параметра масштаба, с увели-

чением масштаба увеличивается шаг сдвига. Это интуитивно понятно, так как для анализа с большим масштабом детали не так важны. Таким образом, $\psi_{m,n}(x) = a_0^{-m/2}\psi(a_0^{-m}x - n)$ — базисные функции вейвлет-преобразования.

По аналогии с терминологией преобразования Фурье, введем ряды вейвлетов непрерывного времени (CTWS):

$$(CTWS_f)_{m,n} = d_{m,n} = \int_{-\infty}^{\infty} a_0^{-m/2}\psi(a_0^{-m}x - n)f(x)dx .$$

Восстановление исходного сигнала из последовательности возможно в том случае, если существуют числа $A > 0$ и $B < \infty$, такие что

$$A\|f(x)\|^2 \le \sum_{m\in Z}\sum_{n\in Z}\left|d_{m,n}\right|^2 \le B\|f(x)\|^2$$

для всех $f(x)$ в $L^2(R)$.

Это означает, что хотя и восстановленный сигнал будет отличаться от исходного, он будет близок к исходному в среднеквадратической норме. Если $A = B = 1$ и $a_0 = 2$, то возможно полное восстановление исходного сигнала и семейство базисных функций $\psi_{a,b}(x)$ образует ортогональный базис. Тогда

$$f(x) = C_\psi \sum_{m\in Z}\sum_{n\in Z} d_{m,n}2^{-m/2}\psi(2^{-m}x - n) .$$

Кратномасштабный анализ

При анализе сигнала зачастую полезно представить сигнал в виде совокупности его последовательных приближений. Например, при передаче изображения по сети можно сначала передать его грубую версию, а затем, исходя из пропускной способности канала, последовательно ее уточнять.

Кратномасштабный анализ обладает целым рядом полезных свойств, главным из которых является возможность выделения из исходного сигнала его деталей различных масштабов. Это позволяет адаптировать кодер к локальным особенностям сигнала и, тем самым, повысить качество кодирования, не уменьшая степени сжатия.

Кратномасштабным анализом называется описание пространства $L^2(R)$ через иерархически вложенные подпространства V_m, которые не пересекаются, и объединение которых дает в пределе все $L^2(R)$, т. е.

$$\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots, \quad \bigcap_{m \in Z} V_m = \{0\}, \quad \bigcup_{m \in Z} V_m = L^2(R).$$

Далее, эти пространства имеют следующее свойство: для любой функции $f(x) \in V_m$, ее «сжатая» версия будет принадлежать V_{m-1},

$$f(x) \in V_m \Leftrightarrow f(2x) \in V_{m-1}.$$

А также, существует такая функция $\phi(x) \in V_0$, что ее сдвиги $\phi_{0,n}(x) = \phi(x-n), n \in Z$ образуют ортонормированный базис пространства V_0.

Из последнего свойства вытекает, что функции $\phi_{m,n}(x) = 2^{-m/2}, \phi(2^{-m}x - n)$ образуют ортонормированный базис пространства V_m. Эти базисные функции называются масштабирующими.

Таким образом, любая функция $f(x) \in L^2(R)$ может быть представлена множеством ее последовательных приближений $f_m(x) \in V_m$. Другими словами, функция $f(x)$ есть предел аппроксимаций $f_m(x) \in V_m$:

$$f(x) = \lim_{m \to -\infty} f_m(x).$$

Из определения кратномасштабного анализа следует, что функция $f_m(x)$ есть ортогональная проекция $f(x)$ на подпространство V_m, то есть

$$f_m(x) = \sum_n \left\langle \phi_{m,n}(x), f(x) \right\rangle \phi_{m,n}(x) = \sum_n c_{m,n} \phi_{m,n}(x).$$

Так как $\phi(x) = \phi_{0,0}(x) \in V_0 \subset V_{-1}$, то

$$\phi_{0,0}(x) = \sqrt{2} \sum_n h_n \phi_{-1,n}(x) = 2 \sum_n h_n \phi\ (2x - n),$$

где h_n — некоторая последовательность.

Это равенство принято называть масштабирующим уравнением. Имеют место следующие соотношения:

$$\sum_n h_n = 1,$$

$$\delta_{0,k} = \left\langle \phi_{0,0}(x), \phi_{0,k}(x) \right\rangle = 2 \sum_n h_n h_{n+2k},$$

и в спектральной области $|H(\omega)|^2 + |H(\omega+\pi)|^2 = 1$, что приводит к $H(0) = 1,\ H(\pi) = 0$.

Введем в рассмотрение еще один объект — пространства W_m:

$$V_{m-1} = V_m \oplus W_m,\ \bigcap_{m\in Z} W_m = \{0\},\ \overline{\bigcup_{m\in Z} W_m} = L^2(R).$$

Пусть $\psi(x) = \psi_{0,0}(x)$ — базисная функция W_0, тогда

$$\psi_{0,0}(x) = \sqrt{2}\sum_n g_n \phi_{-1,n}(x)$$

для некоторой последовательности g_n.

Определим семейство вейвлет-функций $\psi_{m,n}(x) = 2^{-m/2}\psi(2^{-m}x - n)$. Имеют место соотношение: $0 = \langle \phi_{0,0}, \psi_{0,0} \rangle = 2\sum_n h_n g_n$. Легко видеть, что выбор $g_n = (-1)^n = h_{-n+2t+1}$ корректен для всех целых *t*.

Определения вейвлет-функций позволяют записать любую функцию $f(x) \in L^2(R)$ в виде суммы ее проекций на $W_j, j \in R$: $f(x) = \sum_{j=-\infty}^{\infty} e_j(x)$.

Если осуществить анализ до некоторого масштаба *m*, то $f_m(x)$ будет представлена суммой ее грубой аппроксимации $f_m(x) \in V_m$ и множества деталей $e_j \in W_j$:

$$f(x) = f_m(x) + \sum_{j=-\infty}^{m} e_j(x) = \sum_n c_{m,n}\phi_{m,n}(x) + \sum_{j=-\infty}^{m} d_{j,k}\psi_{j,k}(x).$$

Дискретное вейвлет-преобразование

Пусть имеется некоторый дискретный сигнал c_n. Интерпретируем его как коэффициенты разложения некоторой функции $f_0(x) \in V_0$ по базису масштабирующих функций подпространства V_0:

$$f_0(x) = \sum_n c_{0,n}\phi_{0,n}(x), \text{ где } c_{0,n} = c_n,$$

тогда мы можем вычислить аппроксимации этой функции, принадлежащие пространствам $V_1, V_2, \ldots$. Согласно идее кратномасштабного анализа, функция $f_0(x)$ раскладывается на сумму:

$$f_0(x) = f_1(x) + e_1(x) = \sum_k c_{1,k}\phi_{1,k}(x) + \sum_k d_{1,k}\psi_{1,k}(x).$$

Таким образом, мы получили две новые последовательности коэффициентов $c_{1,n}$ и $d_{1,n}$. Этот процесс может быть продолжен разложением $f_1(x), f_2(x), \ldots$, и функция $f_0(x)$ (а следовательно, и исходный дискретный сигнал c_n) будет представлена совокупностью коэффициентов $d_{m,n}, m \in Z^+, n \in Z$. Так определяются дискретные ряды вейвлетов (DTWS).

Учитывая, что масштабирующие функции образуют базисы соответствующих пространств, можно показать, что

$$c_{j,k} = \sqrt{2}\sum_n c_{j-1,n}h_{n+2k} \quad \text{и} \quad d_{j,k} = \sqrt{2}\sum_n c_{j-1,n}g_{n+2k},$$

обратный процесс: $c_{j-1,n} = \sqrt{2}\sum_k c_{j,k}h_{n+2k} + \sqrt{2}\sum_k d_{j,k}g_{n+2k}$.

Последовательности h_n и g_n называются фильтрами. Имеют место следующие соотношения на коэффициенты фильтров:

$$2\sum_k (h_{n+2k}h_{p+2k} + g_{n+2k}g_{p+2k}) = \delta_{n,p},$$

$$2\sum_n h_{n+2k}h_{n+2p} = 2\sum_n g_{n+2k}g_{n+2p} = \delta_{k,p},$$

$$2\sum_n h_{n+2k}h_{n+2p} = 0.$$

На практике мы имеем дело с дискретным сигналом конечной длины. Следовательно, нужно скорректировать вейвлет-преобразование таким образом, чтобы из сигнала какой-то длины получать последовательность вейвлет-коэффициентов той же длины. Получившееся преобразование назовем дискретным вейвлет-преобразованием (DWT).

Обозначим через вектор v^j последовательность конечной длины $c_{j,n}$. Этот вектор преобразуется в вектор v^{j+1}, содержащий последовательности $c_{j+1,n}$ и $d_{j+1,n}$, каждая из которых вполовину короче, чем $c_{j,n}$: $v^{j+1} = M_j v^j$, где M_j — квадратная матрица, состоящая из нулей и элемен-

тов h_n умноженных на $\sqrt{2}$. В силу свойств последовательности h_n, матрица M_j является ортонормированной, следовательно $M_j^T = (M_j)^{-1}$.

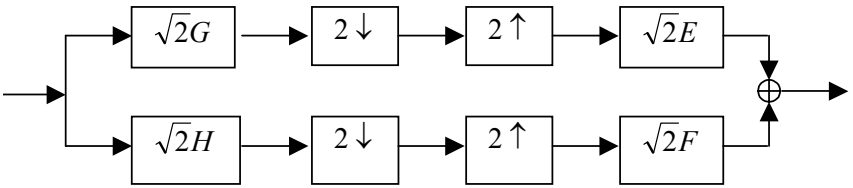
Таким образом, один шаг итерационной процедуры дискретного вейвлет-преобразования для сигнала длины $N = 8$ и фильтра длины $L = 4$ есть:

$$\begin{bmatrix} c_{1,0} \\ c_{1,1} \\ c_{1,2} \\ c_{1,3} \\ d_{1,0} \\ d_{1,1} \\ d_{1,2} \\ d_{1,3} \end{bmatrix} = \sqrt{2} \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & & & & \\ & & h_0 & h_1 & h_2 & h_3 & & \\ & & & & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & & & & & h_0 & h_1 \\ h_3 & -h_2 & h_1 & -h_0 & & & & \\ & & h_3 & -h_2 & h_1 & -h_0 & & \\ & & & & h_3 & -h_2 & h_1 & -h_0 \\ h_1 & h_0 & & & & & h_3 & -h_2 \end{bmatrix} \begin{bmatrix} c_{0,0} \\ c_{0,1} \\ c_{0,2} \\ c_{0,3} \\ c_{0,4} \\ c_{0,5} \\ c_{0,6} \\ c_{0,7} \end{bmatrix}$$

Полное преобразование состоит из $\log_2 N$ итераций.

Следует заметить, что в 4-й и 8-й строках последовательность h_n циклически сдвинута: коэффициенты, выходящие за пределы матрицы, помещены в ту же строку слева. Это означает, что DWT есть точно один период длины N DTWS сигнала $\tilde{c}_n$, получаемого путем бесконечного периодического продолжения сигнала c_n.

В теории вейвлет-преобразований принято описывать применение DWT блок-схемой банка фильтров:



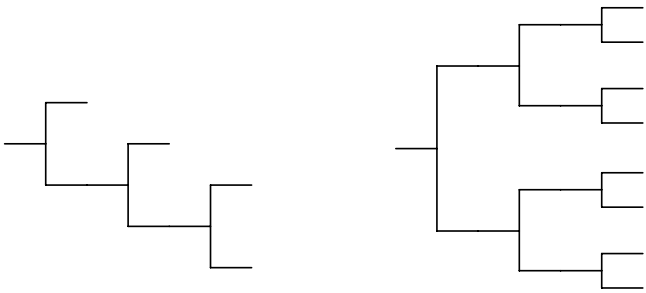
Фильтры G и H означают фильтрацию при помощи g_m и h_n, а фильтры E и F — фильтрацию при помощи g_{-m} и h_{-n} соответственно. Стрелка вниз с коэффициентом 2 обозначает децимацию результата в 2 раза, а стрелка вверх — умножение на 2.

В нижней ветви схемы осуществляется низкочастотная фильтрация. В результате получается аппроксимация исходного сигнала, лишенная деталей низкочастотная субполоса (НЧ). В верхней ветви выделяется высокочастотная составляющая сигнала (ВЧ). Дальнейшее вейвлет-преобразование получается путем рекурсивного применения того же банка фильтров к НЧ составляющей сигнала. При вейвлет-преобразовании изображения, преобразование последовательно применяется и к строкам, и к столбцам (строится так называемая пирамида Маллата).

Адаптивные вейвлет-преобразования

Как уже было сказано выше, вейвлет-преобразование изображения осуществляется путем каскадного соединения банков фильтров по НЧ составляющей сигнала. Такой подход применим при неявном предположении, что основная информация о сигнале содержится в его НЧ области. В общем случае это предположение неверно. Желательно было бы иметь схему способную адаптироваться к конкретным свойствам сигнала. Был разработан целый ряд таких схем, и они получили название «пакет вейвлетов».

Метод пакета вейвлетов основан на определении области, по которой выгоднее производить каскадирование банков фильтров. Для этого сначала проводится каскадирование по обеим полосам. Получается полное, сбалансированное дерево:



вейвлет-декомпозиция полная декомпозиция

Далее, вводится некоторая функция стоимости, на основе которой определяется наилучший путь по этому дереву.

Пакеты вейвлетов были разработаны и изучены Р. Койфманом и М. Викерхаузером. В качестве стоимости они использовали энтропию, понимаемую ими, как концентрацию числа коэффициентов, требующихся для опи-

сания сигнала. Данная функция M будет большой, если коэффициенты примерно одной величины, и малой, если все, кроме нескольких коэффициентов, близки к нулю: $M = e^{-\sum_n p_n \log p_n}$, где $p_n = |x|^2 \|x\|^{-2}$. Энтропия вычисляется для каждого узла полного дерева. Далее, сравнивается энтропия двух потомков и их общего предка на дереве. Если энтропия предка оказалась меньше, то от декомпозиции отказываются. Алгоритм рекурсивно продолжается до достижения вершины дерева.

Для решения задачи сжатия изображений выбор энтропии в качестве функции стоимости, по-видимому, не является наилучшим. В некоторых работах в качестве функции стоимости используется функционал Лагранжа: $J = D + \lambda R$, где D — это искажение (средний квадрат ошибки), вносимое за счет непередачи коэффициента узла, R — количество бит, требуемых для описания коэффициента этого узла, λ — множитель Лагранжа. Алгоритм принятия решения такой же, как и в предыдущем методе. Этот алгоритм получил название одиночного (частотного) дерева.

Вейвлет-пакеты являются более гибким средством, чем вейвлет-преобразование, но, не смотря на это, они не изменяются в пространстве. Поскольку в данной статье рассматривается применение вейвлет-преобразований к кодированию видеопоследовательностей (и изображений, как их частного случая), то имеет смысл рассмотреть алгоритмы, учитывающие изменение исходного сигнала в пространстве. Такие методы получили название алгоритма двойного дерева.

Алгоритм двойного дерева основан на совместном поиске наилучшего разбиения исходного сигнала и в пространственной, и в частотной области. Для простоты, будем предполагать, что исходный сигнал является изображением и в пространственной области он разбивается на некоторые прямоугольные области, размерность которых кратна степени двойки. Далее, для каждой области строится разложение пакетами вейвлетов. Затем значения стоимостей Лагранжа каждой области записываются в виде бинарного дерева. К получившемуся дереву применяется алгоритм одиночного дерева для нахождения наилучшего разбиения исходного сигнала на области.

Алгоритм двойного дерева обладает некоторой асимметричностью. Действительно, деревья в частотной области строятся над пространственными областями, а не наоборот. Этот недостаток можно устранить, построив дерево, в котором кандидатом на дальнейшее разбиение будет являться как пространственная область, так и частотная субполоса. Это дерево (так называемое частотно-пространственное дерево), имеет структуру квадродерева. В самом деле, каждый родительский узел имеет двух пространственных

и двух частотных потомков. Обрезание такого дерева происходит путем сравнения стоимостей Лагранжа: сравниваются пространственные и частотные пары в направлении от листьев к корню дерева. В результате выполнения алгоритма получается оптимальное бинарное дерево разбиения по частоте и пространству полной глубины.

В заключение темы адаптивных вейвлет-преобразований, приведем значения вычислительной сложности всех трех алгоритмов:

Алгоритм	Сложность вычислений
одиночное дерево	$O(Nd)$
двойное дерево	$O(Nd^2)$
пространственно-частотное дерево	$O(N2^d)$

где N — размерность сигнала, d — максимальная высота дерева.

ПРИМЕНЕНИЕ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЙ ДЛЯ КОДИРОВАНИЯ ИЗОБРАЖЕНИЙ

Вейвлет-кодер состоит из трех основных частей: декоррелирующее преобразование, процедура квантования и энтропийное кодирование. Рассмотрим каждую составляющую подробно.

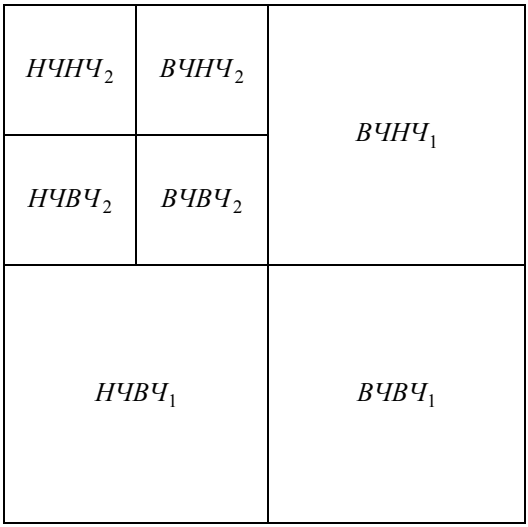
Декоррелирующее преобразование

Выбор оптимального базиса вейвлетов чрезвычайно сложная и вряд ли решаемая задача. Известен ряд критериев построения «хороших» вейвлетов, среди которых наиболее важными являются: гладкость, точность аппроксимации, величина области определения, частотная избирательность. К сожалению, на данный момент наилучшая комбинация этих свойств неизвестна. Более того, в последних работах требование гладкости базиса подвергается сомнению.

Простейшим видом вейвлет-базиса для изображений является разделимый базис, получаемый сжатием и растяжением одномерных вейвлетов. Использование разделимого преобразования сводит проблему поиска эффективного базиса к одномерному случаю, что сильно упрощает задачу, и почти все известные кодеры используют этот подход. Однако неразделимые базисы могут быть более эффективными.

Прототипами базисных функций разделимого преобразования являются функции $\phi(x)\phi(y),\phi(x)(y),(x)\phi(y),(x)(y)$. На каждом шаге преобразования выполняется не одно, а два разбиения по частоте: сначала по строкам, затем по столбцам изображения. Предположим, изображение имеет размерность NxN, тогда после каждого шага преобразования получаются 4 изображения размера $\frac{N}{2}x\frac{N}{2}$ (см. рис.).

Д. Вилласенор систематически протестировал все биортогональные блоки фильтров минимального порядка с длиной фильтра не более 36. Наилучшим фильтром оказался сплайновый фильтр 7/9. Этот фильтр наиболее часто используется в вейвлет-кодерах изображений.



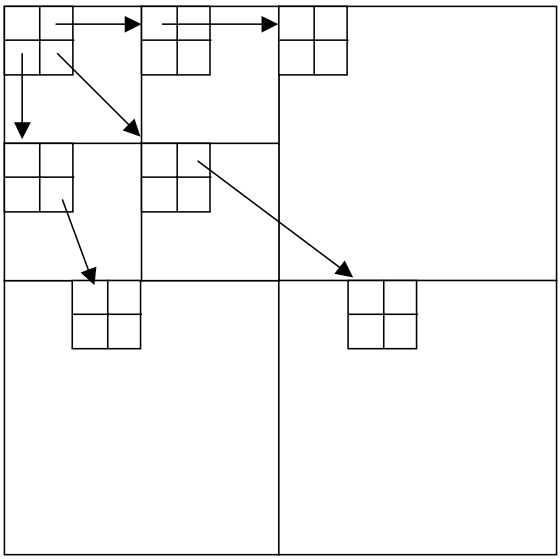
Два уровня вейвлет-преобразования изображения

Далее, традиционное кодирование вейвлет-коэффициентов основано на предположении, что большая часть энергии сосредоточена в малом числе коэффициентов, которые квантуются в соответствии с их значением. Как показывает практика, это предположение бывает неверно.

Традиционное кодирование может быть улучшено путем введения операторов выбора. Вместо квантования коэффициентов в заранее определенном порядке, кодер может выбирать нужные для кодирования элементы

(наиболее значимые). Наиболее значительным результатом этого подхода является разработка алгоритма нуль-дерева и его модификаций.

Впервые идея нуль-дерева была предложена Л. Льюисом и Г. Ноулесом. В их алгоритме применялась древовидная структура данных для описания вейвлет-коэффициентов. Такая структура получается в результате применения двухканального разделимого вейвлет-преобразования. Корневой узел дерева представляет коэффициент в самой НЧ области и имеет три потомка. Узлы дерева соответствуют вейвлет-коэффициентам масштаба, равного их высоте в дереве. Каждый узел имеет четыре потомка, соответствующих вейвлет-коэффициентам следующего уровня и того же пространственного расположения.



Зависимости между коэффициентами вейвлет-преобразования, используемые в алгоритме нуль-дерева

Для каждого из коэффициентов самой НЧ области существуют три таких дерева.

Квантование нуль-дерева основано на наблюдении, что если коэффициент мал, то и все его потомки на дереве зачастую тоже малы. Это объясняется тем, что значимые коэффициенты возникают вблизи контуров и тек-

стур, которые локальны. Дерево или поддерево, которое содержит только незначимые коэффициенты, называется нуль-деревом.

Далее используется следующий итеративный алгоритм квантования. Вначале каждый узел квантуется. Если значение квантованного узла меньше некоторого порога, поддерево, начинающееся с этого узла, объявляется нуль-деревом, и его потомки игнорируются. Эти потомки будут восстановлены декодером как нули. Иначе переходят к четырем потомкам узла, и процедура квантования повторяется. Если узел не имеет потомков, начинает обрабатываться следующий корневой узел.

Особую эффективность этому алгоритму придает совместное кодирование нулей при помощи кодера длин серий (run-length encoder). Для повышения эффективности на вход кодера коэффициенты должны подаваться в определенном порядке, обеспечивающем наибольшие длины последовательностей нулей. Например, в JPEG это зигзагообразное сканирование. Наиболее важным вкладом Льюиса и Ноулеса была демонстрация того, что область вейвлет-коэффициентов хорошо приспособлена для работы такого кодера. В самом деле, генерируются очень длинные последовательности нулей, и нет нужды передавать их длины, поскольку высота дерева известна.

Характеристики этого алгоритма незначительно превосходят JPEG, хотя визуальное качество изображения заметно лучше. Недостатком алгоритма является способ распознавания нуль-дерева. Как было отмечено, если коэффициент мал, то и его потомки малы. В случае если это не так, данный алгоритм ведет к большим искажениям. Преимуществом алгоритма является его вычислительная простота.

Следующее поколение кодеров, в частности алгоритмы Шапиро и Саида–Перельмана, улучшают способ распознавания нуль-дерева.

Шапиро разработал так называемый алгоритм вложенного нуль-дерева (Embedded Zerotree Wavelet encoder — EZW). Он основан на передаче и ненулевых данных, и некоторой карты значений. Если имеется незначащий родительский узел, то очень вероятно, что его потомки тоже будут незначимы. Так что в большинстве случаев генерируется символ нуль-дерева. Если один или больше потомков незначимого узла являются значимыми, то генерируется символ «изолированного нуля». Вероятность этого события ниже, следовательно, для кодирования требуется меньше бит.

Алгоритм EZW генерирует вложенный, иерархический код. Это позволяет осуществить прогрессивную передачу изображения последовательным уточнением при приеме. Подобные коды имеют большой практический интерес по следующим причинам:

1) возможность точного регулирования скорости передачи;
2) возможность восстановления всего изображения при прекращении приема декодером в любой точке. При этом изображение будет максимально хорошего качества для данного числа принятых бит;
3) возможность быстрого просмотра изображения в удаленной базе данных. Для поиска достаточно и грубой копии, а при нахождении нужного изображения оно декодируется полностью.

Алгоритм EZW генерирует вложенный код побитово следующим образом. В начале выполняется частичное упорядочение вейвлет-коэффициентов путем сравнения каждого коэффициента (ВК) с некоторым пороговым значением T. Если ВК < T, то коэффициент считаем значимым, иначе — незначимым. Сканирование производится от НЧ полос к ВЧ. Для кодирования знака и позиции ВК используется двухбитный символ:

- «±» — знак ВК,
- «0» — показывает, что ВК незначащий,
- «корень нуль-дерева» — показывает, что ВК незначащий со всеми своими потомками (ВК — является корнем нуль-дерева),

т.е. используется межполосная, пространственная корреляция ВК. После генерации и передачи карты значений для значащих коэффициентов должны быть переданы биты, уточняющие их значение («карта данных»). Далее карта значений и карта данных сжимаются арифметическим кодером. В том случае, если не исчерпан ресурс скорости передачи, порог T делится на 2 и процесс повторяется.

На начальных итерациях (при большом T) в карте значений будет встречаться много нулей. Роль нуль-дерева заключается в предотвращении передачи лишних нулей. Символ нуль-дерева может снова и снова генерироваться для данного ВК, пока он не станет больше порогового значения, и тогда будет передано квантованное значение ВК.

А. Саид и В. Перельман улучшили алгоритм EZW. Их версия кодера получила название «установка подразделений в иерархических деревьях» (Set Partition In Hierarchical Trees — SPIHT). Он основан на обнаружении схожих фрагментов в различных поддеревьях вейвлет-коэффициентов.

Квантование

Обычно в алгоритмах кодирования изображений используется скалярное квантование. Числовая прямая разбивается на отрезки равной длины (неравномерная сетка порождает неравномерное скалярное квантование).

Далее, квантуемый элемент, попадающий в какой-то отрезок, заменяется центром этого отрезка.

Более эффективным методом квантования является векторное квантование. Этот алгоритм состоит из двух этапов: составления кодовой книги и собственно кодирования. В кодовой книге хранятся взвешенные комбинации двумерных блоков коэффициентов. Каждому блоку приписан его код, получаемый посредством алгоритма Хаффмана. Кодирование состоит из нахождения для каждого кодируемого блока ближайшего блока из кодовой книги относительно некоторой меры, например среднеквадратичной разности. В выходной поток записываются соответствующие коды ближайших блоков. Недостаток этого метода в том, что он требует хранения кодовой книги. Однако, если кодовая книга составлена из элементов самого изображения, то затраты на передачу кодовой книги — минимальны.

Энтропийное кодирование

Субоптимальное энтропийное кодирование можно реализовать при помощи алгоритма арифметического кодирования. Кодеру требуется оценить распределение квантованных коэффициентов. Эта оценка получается путем аппроксимации распределения коэффициентов гауссовской плотностью и вычислением параметров распределения. Оценка параметров может производиться и в процессе работы, «на ходу». Такой подход позволяет учитывать локальные изменения статистики изображения.

Ввиду того, что изображение не является случайным гауссовским источником, коэффициенты преобразования обладают определенной структурой. Энтропийный кодер может каким-то образом использовать эту структуру, осуществляя предсказание параметров. В ряде работ отмечалось, что это приводит к незначительному улучшению эффективности.

На практике вместо арифметического кодера используется кодер Хаффмана. Причина заключается в меньшей вычислительной сложности алгоритма Хаффмана, а также в том, что большинство вариаций алгоритма арифметического кодирования запатентованы. Однако, поскольку патент имеет ограниченный срок действия (а у большей части патентов этот срок истек в 2001 году), в скором времени возможно появление целой серии кодеров, использующих арифметическое кодирование.

ЗАКЛЮЧЕНИЕ

Современные исследования в области сжатия изображений ведутся по разным направлениям: нелинейные вейвлет-преобразования, помехозащищенные кодеры и т.д.

Особый интерес представляет адаптация вейвлет-кодирования изображений для кодирования видео. Здесь можно сочетать внутрикадровое кодирование с межкадровым предсказанием, аналогично стандарту MPEG4. Можно также рассматривать видеопоследовательность как трехмерный сигнал и применять трехмерный вейвлет-анализ. Здесь возникают трудности с представлением получающейся древовидной структуры, но работы в этом направлении активно ведутся.

В заключение стоит отметить, что вейвлеты и сопутствующие им идеи внесли существенный вклад в теорию и практику кодирования изображений и, по-видимому, будут оставаться основным направлением исследований в этой области в ближайшем будущем.

СПИСОК ЛИТЕРАТУРЫ

1. **Воробьев В.И., Грибунин В.Г.** Теория и практика вейвлет-преобразования. — СПб: ВУС, 1999.
2. **Mallat S.** A Theory for Multiresolution Signal Decomposition: The Wavelet Representation // IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989. — Vol. 11. — P. 674–693.
3. **Shapiro J.** Embedded Image Coding Using Zerotrees of Wavelet Coefficients // IEEE Transactions on Signal Processing, 1993. — Vol. 41, No. 12.
4. **Said A., Pearlman W.** A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees // IEEE Transactions on Circuits and Systems for Video Technology, 1996. — Vol. 6. — P. 243–250.
5. **Antonini M., Barlaud M., Mathieu P., Daubechies I.** Image coding using wavelet transform // IEEE Transactions On Image Processing, 1992. — Vol. 1, № 2. — P. 205–220.

В.Н. Касьянов, И.Л. Мирзуитова

УПОРЯДОЧЕННЫЕ ДИАГРАММЫ БИНАРНЫХ РЕШЕНИЙ*

ВВЕДЕНИЕ

Многие проблемы в таких областях, как проектирование цифровых систем, комбинаторная оптимизация, математическая логика и искусственный интеллект, могут быть сформулированы в терминах операций над небольшими конечными областями. Посредством введения бинарного кодирования элементов этих областей, упомянутые проблемы могут быть сведены к операциям над булевыми значениями. Используя символьное представление булевых функций, мы можем выражать проблемы в обобщенной форме. Решение такой обобщенной проблемы с помощью символьной обработки булевых функций дает возможность разрешить большое количество конкретных частных случаев. Следовательно, эффективный метод представления и обработки булевых функций может привести к решению большого класса сложных проблем.

Упорядоченные бинарные диаграммы решений (OBDD) представляют булевы функции в виде ориентированных ациклических графов. Они образуют каноническое представление, с помощью которого проверка таких функциональных характеристик, как выполнимость и эквивалентность, может быть произведена простым образом. Множество операций на булевых функциях может быть реализовано в виде графовых алгоритмов на OBDD-структурах данных. Используя упорядоченные бинарные диаграммы решений, мы можем решать проблемы различных классов с помощью *символьного анализа,* если все возможные вариации параметров системы и технических условий можно закодировать с помощью булевых переменных.

Данная работа представляет собой обзор символьных булевых операций с упорядоченными бинарными диаграммами решений. Описываются OBDD-диаграммы и алгоритмы их построения и обработки. Рассматриваются некоторые базовые техники для представления и обработки таких математических структур, как функции, множества и отношения, с помощью символьных булевых операций. Эти техники проиллюстрированы описани-

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

ем некоторых приложений, использующих OBDD, а также рассмотрены возможные направления дальнейших исследований. Хотя большая часть описанных приложений относится к области проектирования цифровых систем, похожие методы могут быть применены и в других областях.

1. УПОРЯДОЧЕННЫЕ ДИАГРАММЫ БИНАРНЫХ РЕШЕНИЙ И ЛОГИЧЕСКИЕ ФРАГМЕНТЫ

Бинарная диаграмма решений (BDD) представляет булеву функцию в виде корневого ориентированного ациклического графа — дэга с одной выделенной вершиной, называемой корнем, которая не имеет предшественников и из которой достижимы все его вершины.

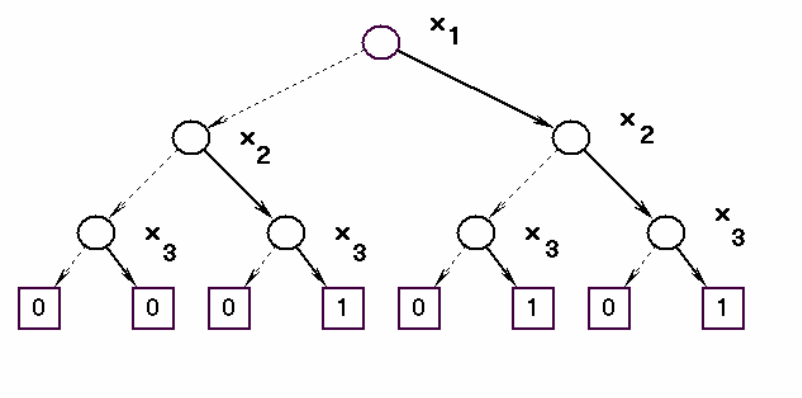


Рис. 1. Бинарная диаграмма решений булевой функции *f* (табл. 1), являющейся деревом. Пунктирные и сплошные дуги соответствуют случаям, когда значение разрешающей переменной равно 0 или 1 соответственно

На рис. 1 показан пример представления функции $f(x_1,x_2,x_3)$, определяемой табл. 1, когда указанный граф является деревом. Каждая его нетерминальная (не висячая) вершина графа v помечена переменной ПЕР(v) и имеет две исходящих дуги к двум сыновьям: ЛЕВ(v) (на рисунке показана пунктирной линией) в случае, когда переменной присваивается значение 0, и ПРАВ(v) (показана сплошной линией) в случае, когда переменной присваивается значение 1. Каждая терминальная (висячая) вершина графа помечена либо 0, либо 1. Для заданного распределения значений истинности логических переменных значение функции, реализованной диаграммой, определя-

ется с помощью продвижения по пути от корня к терминальной вершине по ветвям, соответствующим присваиваемым переменным значениям. В качестве значения функции берется метка найденной терминальной вершины. Дуги на данном рисунке упорядочены таким образом, что значения терминальных вершин (слева направо) соответствуют их вхождениям в таблицу истинности (сверху вниз).

Таблица 1

Табличное задание функции f

X_1	X_2	X_3	F
0	0	0	0
0	0	1	0
0	0	0	0
0	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
1	1	1	1

В случае упорядоченной бинарной диаграммы решений (OBDD) рассматривается некоторый линейный порядок < на множество переменных и требуется, чтобы для любой вершины *u* и любого ее преемника *v,* не являющегося терминальной вершиной, соответствующие им переменные были упорядочены: ПЕР*(u)* < ПЕР*(v)*. К примеру, в дереве решений на рис. 1 переменные упорядочены следующим образом: $x_1 < x_2 < x_3$.

Теоретически порядок переменных может быть выбран произвольно — алгоритмы будут работать корректно при любом выборе порядка. Но на практике выбор подходящего упорядочения является критичным для эффективности символьной обработки. Этот вопрос подробнее рассматривается ниже.

Определяются три правила преобразования описанных графов, не изменяющих значение представляемой функции.

Удаление дублирующих терминальных переменных. Удаляются все терминальные вершины с заданной меткой, кроме одной, а все дуги, ведущие к удаленным вершинам, перенаправляются в оставшуюся вершину.

Удаление дублирующих нетерминальных вершин. Если для нетерминальных вершин *u* и *v* выполняются следующие условия: ПЕР(*u*)=ПЕР(*v*), ЛЕВ(*u*)=ЛЕВ(*v*) и ПРАВ(*u*)=ПРАВ(*v*), то одна из этих вершин удаляется, а все дуги, входящие в нее, перенаправляются во вторую вершину.

Удаление избыточных проверок Если для нетерминальной вершины *v* верно ЛЕВ(*v*)=ПРАВ(*v*), то *v* удаляется, а все дуги, входящие в нее, перенаправляются в ЛЕВ(*v*).

Начиная с любой бинарной диаграммы решений, обладающей порядком, можно пытаться сократить ее размер путем последовательного применения правил преобразования. Термин «OBDD» используется для обозначения максимально приведенного графа, обладающего некоторым фиксированным порядком на множестве переменных. Пример на рис. 2 иллюстрирует процесс приведения дерева решений, представленного на рис. 1, к OBDD. Применение первого правила преобразования сокращает количество терминальных вершин с 8 до 2. Применение второго правила удаляет две вершины с переменной x_3 и дуги к терминальным вершинам с метками 0 (ЛЕВ) и 1 (ПРАВ). Применение третьего правила удаляет две вершины: одну с переменной x_3 и одну с переменной x_2. В общем случае правила преобразования должны применяться неоднократно, поскольку каждое применение преобразования может привести к появлению новых возможностей для других преобразований.

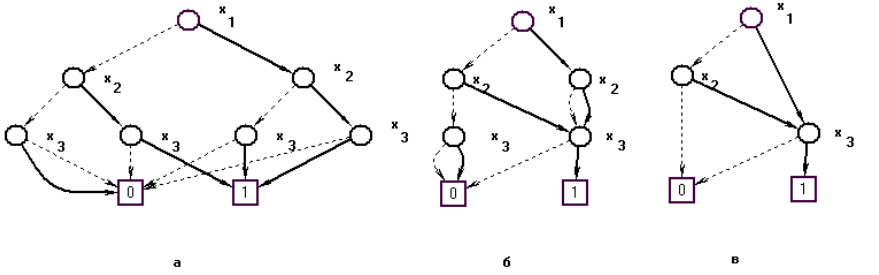


Рис. 2. Приведение дерева решений к OBDD.
Применение трех правил преобразования к дереву на рис. 1 дает в результате каноническое представление функции в виде OBDD: удаление дублирующих терминальных переменных (а), удаление дублирующих нетерминальных переменны (б), удаление избыточных проверок (в)

Представление функции в виде OBDD является универсальным и каноническим.

Свойство А. *Любая булева функция при любом заданном порядке на переменных имеет OBDD-представление.*

Свойство Б. *Для любого заданного порядка на переменных две упорядоченные бинарные диаграммы решений одной и той же функции изоморфны.*

Из этого вытекают несколько важных следствий. Легко может быть проверена функциональная эквивалентность. Функция является выполнимой в том и только том случае, когда ее OBDD-представление не сводится к одиночной терминальной вершине с меткой 0. Любая тавтологическая функция должна иметь терминальную вершину с меткой 1 в качестве своего OBDD-представления. Если функция не зависит от переменной *x,* то ее OBDD-представление не может иметь вершин, помеченных *x*. Таким образом, как только построены представления функций в виде OBDD, многие их характеристики становятся легко проверяемыми.

Как демонстрируют рис. 1 и 2, можно строить OBDD-представление для функции, заданной таблично, построив и затем сократив ее дерево решений. Однако этот подход является практичным только для функций с небольшим количеством переменных, поскольку и таблица истинности, и дерево решений имеют экспоненциальный по отношению к числу переменных размер. Поэтому OBDD обычно строятся при помощи «символьного выполнения» логического выражения или сети логических элементов с применением операции APPLY, описанной в п. 1.2.

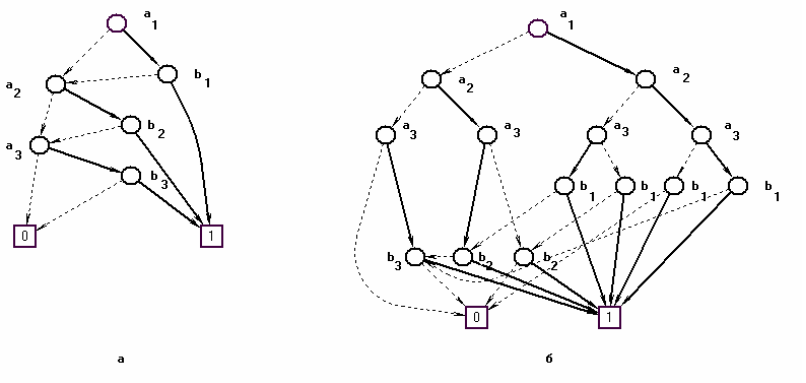


Рис. 3. OBDD-представления одной и той же функции для двух различных способов упорядочения переменных

Форма и размер OBDD, представляющей функцию, зависит от способа упорядочения переменных. На рис. 3 приведен пример двух OBDD-представлений функции, заданной булевым выражением

$$a_1*b_1 + a_2*b_2 + a_3*b_3,$$

где «*» обозначает операцию AND, а «+» — операцию OR. Для примера на рис. 3,а был задан следующий порядок переменных:

$$a_1 < b_1 < a_2 < b_2 < a_3 < b_3,$$

тогда как на рис. 3, б порядок был таким:

$$a_1 < a_2 < a_3 < b_1 < b_2 < b_3.$$

Можно обобщить эту функцию следующим образом:

$$a_1*b_1 + a_2*b_2 + \ldots + a_n*b_n.$$

Обобщение первого способа упорядочения переменных до

$$a_1 < b_1 < \ldots < a_n < b_n$$

дает OBDD-представление с количеством нетерминальных вершин, равным $2n$. Обобщение второго способа до

$$a_1 < \ldots a_n < b_1 < \ldots < b_n,$$

с другой стороны, дает в результате OBDD-представление с $2(2^n - 1)$ числом нетерминальных вершин. Для больших значений n разница между линейным возрастанием первого представления и экспоненциальным ростом второго будет иметь существенное влияние на объем требуемой памяти и эффективность алгоритма обработки.

Исследуя структуру двух графов на рис. 3, мы видим, что в первом случае переменные объединены в пары в соответствии с их вхождением в булево выражение

$$a_1*b_1 + a_2*b_2 + a_3*b_3.$$

Таким образом, для любого второго уровня в графе имеются только два возможных пункта назначения исходящих ветвей: одна ведет к терминальной вершине, помеченной 1, в случае, когда соответствующее произведение дает 1; вторая ведет к следующему уровню в случае, если все произведения переменных до этого момента были равны 0. С другой стороны, все три уровня во втором случае образуют полное бинарное дерево, описывающее все возможные присваивания переменным a. В общем случае при каждом

присваивании переменным *a* значение функции единственным образом зависит от присваивания переменным *b*. Поскольку мы обобщаем эту функцию и порядок на более чем 2*n* переменных, первые *n* уровней OBDD образуют полное бинарное дерево.

Большинство приложений, использующих OBDD, выбирают некоторый порядок переменных с самого начала и строят все графы в соответствии с этим порядком. Поскольку проблема нахождения оптимального упорядочения переменных для OBDD является *NP*-трудной, порядок, как правило, выбирается либо вручную, либо с помощью эвристического анализа представляемой системы. Например, были разработаны некоторые эвристические методы, которые, получив на входе сеть логических элементов, обычно выводят хороший способ упорядочения переменных, представляющих первичные входные данные. Другие методы были разработаны для анализа последовательных систем. Заметим, что упомянутые эвристики не обязаны находить оптимальное упорядочение — выбор порядка не влияет на корректность результатов. Как только обнаружен порядок, не приводящий к экспоненциальному росту, операции на OBDD показывают приемлемую эффективность.

OBDD представляет собой практичный подход к символьной булевой обработке только в том случае, когда размер графа остается значительно меньше экспоненциального относительно количества переменных. Как показано в предыдущем примере, некоторые функции чувствительны к упорядочению переменных, но остаются вполне компактными, если выбран хороший порядок. Далее, имеется достаточный эмпирический опыт, чтобы утверждать, что многие функции из реальных приложений могут быть эффективно представлены в виде OBDD. Одним из способов более полно представить сильные и слабые стороны OBDD-представления может быть вывод нижних и верхних границ для важнейших классов булевых функций.

В табл. 2 сведены воедино асимптотические оценки степени возрастания для нескольких классов булевых функций и их чувствительность к упорядочению переменных. Симметричные функции, у которых значение функции зависит только от количества аргументов, равных 1, нечувствительны к порядку переменных. За исключением тривиального случая константных функций, графы симметричных функций попадают в промежуток между линейными (например, четность) и квадратичными (например, не менее половины входных данных равны 1).

Таблица 2

Сложность OBDD-представления для широко распространенных классов функций

Класс функций	Сложность	
	Лучший случай	Худший случай
Симметричные	Линейная	Квадратичная
Целочисленное сложение (любой бит)	Линейная	Экспоненциальная
Целочисленное умножение (средние биты)	Экспоненциальная	Экспоненциальная

Мы можем рассматривать каждое выходное значение n-битового сумматора как булеву функцию на переменных

$$a_0 , a_1 , ..., a_{n-1},$$

представляющих один операнд, и

$$b_0 , b_1 , ..., b_{n-1},$$

представляющих второй операнд. Функция для любого бита имеет OBDD-представление линейной сложности для порядка

$$a_0 < b_0 < a_1 < b_1 < ... < a_{n-1} < b_{n-1}$$

и экспоненциальной сложности для порядка

$$a_0 < a_1 < ... < a_{n-1} < b_0 < b_1 < ... < b_{n-1}.$$

Фактически представление таких функций похоже на представление функции, изображенной на рис. 3.

Булевы функции, представляющие целочисленное умножение, являют собой особенно сложный случай для OBDD.

Свойство В. *Вне зависимости от способа упорядочения булева функция, представляющая любое из двух средних выходных значений n-битного умножителя, имеет экспоненциальное OBDD-представление.*

Верхние границы для других классов булевых функций могут быть выведены на основе структурных свойств их логических сетей. Рассмотрим

сеть с *n* первичными входными значениями и одним первичным выходным значением, состоящую из m «логических блоков». Каждый блок может иметь несколько входов и выходов. Первичные данные представлены «исходными» блоками без входа и с одним выходом. В качестве примера на рис. 4 приведена сеть, дающая в качестве выходного значения самый важный бит суммы *n*-битного сумматора. Эта сеть состоит из цепи переноса, последнее значение которой вычисляется из c_{n-1}. Блоки, помеченные «2/3», вычисляют функцию MAJORITY, которая дает на выходе 1, если по меньшей мере два входных значения равны 1. Выходное значение вычисляется как EXCLUSIVE-OR самых важных битов входных значений и c_{n-1}.

Определим *линейное расположение* сети как нумерацию блоков от 1 до *m* таким образом, что блок, вычисляющий выходное значение, пронумерован последним. Определим *прямое поперечное сечение* в блоке *i* как общее количество проводов, ведущих из выхода блока *j (j < i)* ко входу блока *k (i ≤ k)*. Определим прямое поперечное сечение цепи w_f (относительно расположения) как максимальное прямое поперечное сечение среди всех блоков. Как видно на рис. 4, прямое поперечное сечение нашей цепи сумматора равно 3 (обозначено пунктирной линией). Подобным же образом определим *обратное поперечное сечение* в блоке i как общее количество проводов, ведущих из выхода блока *j (j > i)* ко входу блока *k (i ≥ k)*. В тех расположениях, где блоки упорядочены *топологически*, обратное поперечное сечение равно 0. На рис. 4 мы имеем именно такую картину. Определим прямое поперечное сечение цепи w_r (относительно расположения) как максимальное прямое поперечное сечение среди всех блоков. Можно показать, что существует OBDD-представление функции цепи с не более чем $n2^{w_f 2^{w_r}}$ вершинами. Более того, нахождение расположения с низкими поперечными сечениями означает нахождение хорошего способа упорядочения переменных функции — таким порядком будет инверсия порядка соответствующих исходных блоков в их расположении.

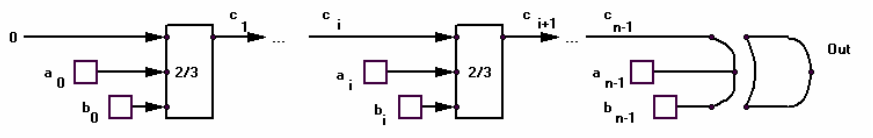


Рис. 4. Линейное расположение цепи,
вычисляющей самый главный бит целочисленного сложения

С помощью сетевых реализаций можно найти подходящие границы для различных булевых функций. Например, функции, имеющие реализацию с постоянным прямым поперечным сечением и нулевым обратным поперечным сечением (такие, как цепь сумматора на рис. 4), имеют линейное OBDD-представление. Симметричная функция от n переменных может быть реализована в виде цепи с прямым поперечным сечением 2 + log *n* и обратным поперечным сечением 0. Эта цепь состоит из серии каскадов, вычисляющих общее количество входных значений, равных 1, кодируя их в виде $[\log_2 n]$-битного двоичного числа. Эта реализация приводит к квадратичной верхней границе OBDD-представления, указанной в табл. 1.

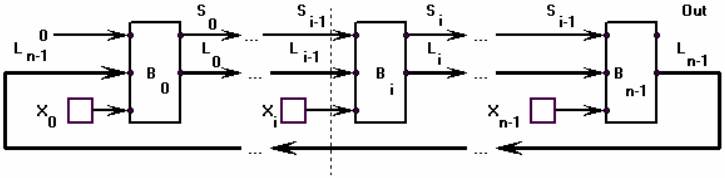


Рис. 5. Линейное расположение кольцевой цепи для *Within-K*. Пунктирной линией показано прямое поперечное сечение $2 + [\log_2 K]$ и обратное поперечное сечение $[\log_2 K]$

На рис. 5 изображено применение этого результата к цепи с ненулевым обратным поперечным сечением. Эта цепь представляет общую реализацию функции *Within-K*, где *K* — некоторая константа, такая, что $0 < K < n$. Для входных значений

$$x_0, x_1, \ldots, x_{n-1}$$

функция выдает значение 1 в случае, когда имеются входные значения x_i и $x_{i'}$, равные 1, причем $i' = i + j \bmod n$ для некоторого значения j, $0 < j < K$. Как показано на рис. 5, эта функция может быть вычислена с помощью серии блоков, расположенных в виде кольца, где каждый блок B_i выдает на выходе 1-битное значение s_i и k-битное целочисленное значение L_i, где $k = [\log_2 K]$:

$$s_i = \begin{cases} 1, \text{ если } x_i = 1 \text{ и } L_{i-1} \neq 0, \\ s_{i-1}, \text{ в противном случае,} \end{cases}$$

$$L_i = \begin{cases} K-1, \text{если } x_1 = 1, \\ L_{i-1} - 1, \text{ если } x_1 = 0 \text{ и } L_{i-1} > 0, \\ 0, \text{ в противном случае.} \end{cases}$$

В данной реализации каждый сигнал L_i кодирует количество оставшихся позиций, с которыми может составить пару самое последнее входное значение 1, в то время как каждый сигнал s_i показывает, встретилась ли уже пара входных значений, равных 1, внутри дистанции K. Чтобы удовлетворять условию близости модулей, выход L_{i-1} последнего каскада направляется на вход начального каскада. Заметим, что, хотя эта цепь имеет циклическую структуру, ее выходное значение однозначно определяется входными значениями. Как отмечено пунктирной линией, эта кольцевая структура может быть «сплющена» до линейного расположения с прямым поперечным сечением $k + 2$ и обратным поперечным сечением k. Эта конструкция дает верхнюю границу OBDD-представления, равную $(8K4^K)n$. Для константных значений K размер OBDD будет линейным, хотя постоянный коэффициент будет быстро возрастать с ростом K. Существует обобщение этой техники до *древовидного расположения*, в котором сеть организована в виде дерева логических блоков с коэффициентом ветвления b и первичным выходным значением, производимым корневым блоком. При таком расположении прямое поперечное сечение соответствует количеству проводов, направленных к корню (для обратного сечения — соответственно, из корня). Эта конструкция дает верхнюю границу размера OBDD-представления

$$n[2^b n^{b-1}]^{w_f 2^{w_r}}.$$

Верхняя граница для линейного расположения получается подстановкой $b = 1$ в эту формулу. Заметим, что для константных значений b, w_f и w_r размер OBDD будет полиномиальным относительно n.

Эти оценки верхней границы дают некоторое представление о том, какое количество функций, встречающихся в приложениях в области логического проектирования, имеют эффективные OBDD-представления. Они также позволяют предложить стратегию обнаружения хороших способов упорядочения переменных с помощью нахождения сетевых реализаций с низкими поперечными сечениями. Результаты, полученные для этой формы представления булевых функций, могут оказаться полезными при оценке потенциала OBDD-представления в других областях применения.

Есть целый ряд различных усовершенствований базовой структуры OBDD, дающих значительную экономию при выделении памяти — как правило, наиболее критичного ресурса при определении сложности решаемых проблем (приложения, требующие генерации более чем миллиона вершин OBDD, не так уж легко исполнить на обычных рабочих станциях). К ним относятся, например, использование единственного многокорневого

графа для представления всех требуемых функций, снабжение дуг метками для обозначения булева вычитания и обобщение понятия областей с конечным числом состояний.

2. КОНСТРУИРОВАНИЕ И МАНИПУЛЯЦИЯ

Введем систему обозначений для описания операций на булевых функциях. Мы будем использовать стандартные операции булевой алгебры: + для OR, · для AND, $\oplus$ для EXCLUSIVE-OR и верхний подчерк для NOT. Дополнительно мы будем использовать символ $\overline{\oplus}$, обозначающий дополнение к операции EXCLUSIVE-OR (иногда называемое EXCLUSIVE-NOR). Мы также используем понятия суммы (Σ) и произведения (Π), подразумевая под ними булеву сумму (OR) и произведение (AND). Заметим, что эти операции определяются над *функциями*, также как и над булевыми значениями 0 и 1. К примеру, если f и g — функции над некоторым множеством переменных, то $f + g$ также является функцией (скажем, h) над этими же переменными. Для некоторого присваивания значениям переменных $\vec{a}$, $h(\vec{a})$ выдает значение 1 в том и только том случае, когда либо $f(\vec{a})$, либо $g(\vec{a})$ выдает значение 1. Константные функции, значение которых равно 1 или 0 для любых присваиваний, обозначаются как **1** и **0** соответственно.

Функция, которая получается, когда некоторому аргументу x функции f присваивается константное значение k (0 либо 1), называется *рестрикцией* функции f (иногда ее также называют «кофактором» f) и обозначается $f|_{x\leftarrow k}$. Если даны две рестрикции функции относительно одной переменной, функция может быть реконструирована следующим образом:

$$f = \overline{x} \cdot f|_{x\leftarrow 0} + x \cdot f|_{x\leftarrow 1}.$$

Это тождество часто называют *Шенноновым разложением f* по переменной x, хотя впервые оно было упомянуто Булем.

Многие другие полезные операции могут быть определены в терминах алгебраических операций и операции рестрикции. Операция *композиции*, при которой функция g заменяет переменную x функции f, задается следующим тождеством:

$$f|_{x\leftarrow g} = \overline{g} \cdot f|_{x\leftarrow 0} + g \cdot f|_{x\leftarrow 1}.$$

Операция *квантификации переменной*, при которой некоторая переменная x экзистенциально или универсально оценивается относительно функции f, задается тождествами

$$\exists x\ f = f\big|_{x\leftarrow 0} + f\big|_{x\leftarrow 1},$$
$$\forall x\ f = f\big|_{x\leftarrow 0} \cdot f\big|_{x\leftarrow 1}.$$

Некоторые исследователи предпочитают называть эти операции *сглаживанием* (экзистенциальная квантификация) и *консенсусом* (универсальная квантификация), чтобы подчеркнуть тот факт, что они являются операциями над булевыми функциями, а не над истинностными значениями.

Большое количество символьных операций над булевыми функциями может быть реализовано в виде графовых алгоритмов, применяемых к упорядоченным бинарным диаграммам решений. Эти алгоритмы обладают важным свойством замыкания — если аргументы функции представляют собой OBDD с некоторым порядком, то результатом функции будет OBDD с тем же порядком. Следовательно, мы можем реализовать сложную обработку с помощью последовательности простых манипуляций, всегда работая с OBDD с привычным порядком. Пользователи могут рассматривать библиотеку подпрограмм обработки бинарных диаграмм решений как реализацию абстрактного типа данных — булевой функции. За исключением выбора способа упорядочения переменных, все операции реализуются полностью автоматически. Пользователю нет необходимости вдаваться в детали представления или реализации.

2.1. Операция APPLY

Операция APPLY генерирует булевы функции путем применения алгебраических операций к другим функциям. Если в качестве аргументов заданы функции f и g и бинарный булев оператор <op> (например, AND или OR), то функция APPLY выдает в результате функцию f <op> g. Эта операция является важнейшей для символьной булевой обработки. С ее помощью мы можем находить дополнение функции f, вычисляя $f \oplus 1$. Пусть даны функции f и g и состояние безразличности, выраженное функцией d (то есть $d(\vec{x})$ равно 1 для тех присваиваний переменным $\vec{x}$, для которых значения функции не важны), тогда мы можем проверить, являются ли f и g эквивалентными для всех «небезразличных» состояний, вычисляя значение $(f \overline{\oplus}\ g) + d$ и проверяя, равен ли результат функции **1**. Мы также можем построить OBDD-представление выходных функций сети комбинационно-логических элементов путем «символической интерпретации» сети. Это

производится следующим образом. Вначале мы представляем функцию на каждом первичном входе в виде OBDD, состоящего из проверки одной переменной. Затем в порядке, определяемом структурой сети, мы применяем операцию APPLY для построения OBDD-представления каждого выходного элемента в соответствии с операцией, выполняемой элементом, и OBDD, построенными для входных значений этого элемента.

Алгоритм APPLY производит обход графов аргументов в глубину, поддерживая две хэш-таблицы: одну — для улучшения производительности вычисления, вторую — для построения максимально приведенного графа. Заметим, что в отличие от более ранних алгоритмов, которым требовался отдельный шаг для приведения сокращаемого графа к канонической форме, описанный ниже алгоритм строит приведенную форму сразу. Чтобы проиллюстрировать эту операцию, мы рассмотрим применение операции + к функциям $f(a,b,c,d) = (a + b) \cdot c + d$ и $g(a,b,c,d) = (a \cdot \bar{c}) + d$, OBDD-представление которых показано на рис. 6.

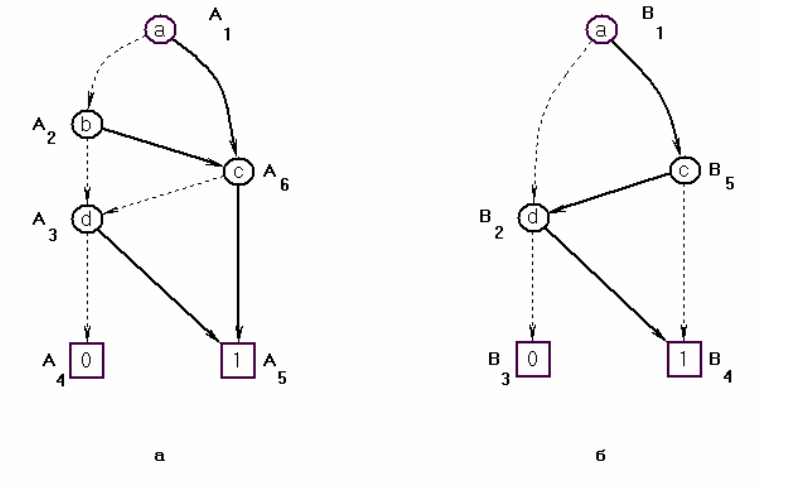


Рис. 6. Аргументы операции APPLY.
Вершины помечены для идентификации их в следе исполнения

Реализация операции APPLY опирается на то, что алгебраические операции связаны с Шенноновым разложением по любой переменной x:

$$f \langle op \rangle g = \bar{x} \cdot (f|_{x \leftarrow 0} \langle op \rangle g|_{x \leftarrow 0}) + x \cdot (f|_{x \leftarrow 1} \langle op \rangle g|_{x \leftarrow 1}). \qquad (1)$$

Заметим, что для функции f, представленной диаграммой с корневой вершиной r_f, рестрикция относительно переменной x, такой, что $x \le \text{ПЕР}(r_f)$, может быть вычислена следующим простым способом:

$$f|_{x\leftarrow 0} = \begin{cases} r_f, & \text{если } x < \text{ПЕР}(r_f), \\ \text{ЛЕВ}(r_f), & \text{если } x = \text{ПЕР}(r_f) \text{ и } b = 0, \\ \text{ПРАВ}(r_f), & \text{если } x = \text{ПЕР}(r_f) \text{ и } b = 1. \end{cases}$$

Т. е. рестрикция представлена либо тем же самым графом, либо одним из подграфов корневой вершины.

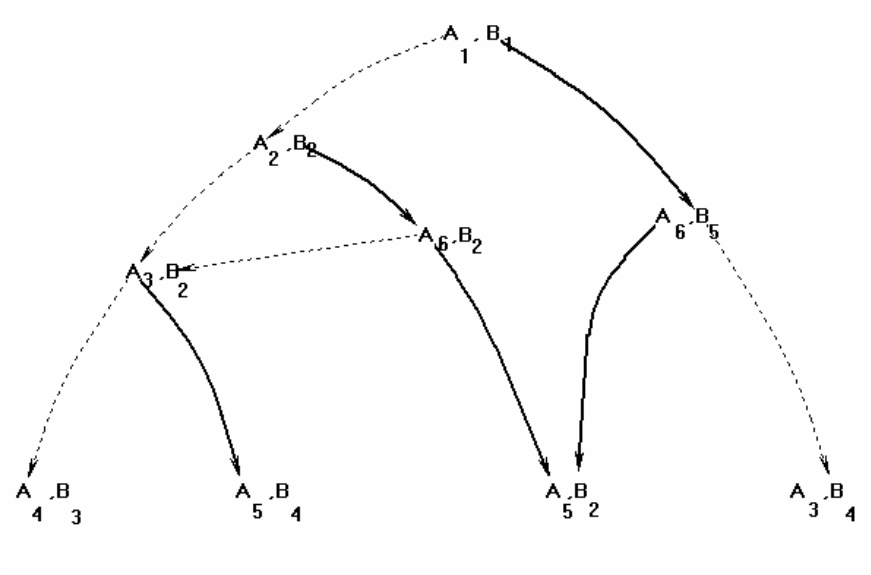


Рис. 7. След исполнения операции APPLY с операцией +.
Каждый шаг выполнения имеет в качестве операндов вершину
из каждого графа-аргумента

Равенство 1 образует базис для рекурсивной процедуры вычисления OBDD-представления f <op> g. Структура рекурсивного вычисления для нашего примера представлена на рис. 7. Заметим, что каждый шаг вычисления идентифицируется вершиной каждого из графов-аргументов. Предположим, что функции f и g представлены OBDD-диаграммами с корневыми вершинами r_f и r_g, соответственно. В случае, когда и r_f и r_g являются терми-

нальными вершинами, рекурсия завершается, возвращая подходящим образом помеченную терминальную вершину. В нашем примере это происходит при вычислении A_4,B_3 и A_5,B_4. В противном случае будем считать *x расщепляющей переменной*, определяя ее как минимум из переменных ПЕР(r_f) и ПЕР(r_g). OBDD-диаграммы для функций

$$f\big|_{x\leftarrow 0} <\text{op}> g\big|_{x\leftarrow 0} \text{ и } f\big|_{x\leftarrow 1} <\text{op}> g\big|_{x\leftarrow 1}$$

строятся путем рекурсивного выполнения рестрикций f и g для значения 0 (обозначенных пунктирными линиями на рис. 7) и для значения 1 (обозначенных сплошными линиями). В нашем примере начальное вычисление для вершин A_1,B_1 вызывает рекурсивное вычисление для вершин A_2,B_2 и A_6,B_5.

Чтобы более эффективно реализовать операцию APPLY, можно ввести два усовершенствования в описанную выше процедуру.

Во-первых, если встречается условие, в котором один из аргументов является терминальной вершиной, представляющей «доминантное» значение для операции <op> (например, 1 для OR или 0 для AND), то можно остановить рекурсию и вернуть подходящим образом помеченную терминальную вершину. В нашем примере это происходит при вычислении для A_5,B_2 и A_3,B_4.

Во-вторых, можно избегать любых множественных рекурсивных вызовов одной и той же пары аргументов путем поддержки хэш-таблицы, в которой каждая запись в качестве ключа содержит пару вершин двух графов-аргументов, а в качестве элемента данных — вершину генерируемого графа. В начале выполнения для аргументов u и v проверяется наличие записи с ключом $<u, v>$ в хэш-таблице. Если такая запись обнаружена, возвращается элемент данных этой записи, что позволяет прекратить дальнейшую рекурсию. Если записи не найдено, производятся описанные выше действия по созданию новой записи перед возвратом результата. В нашем примере с помощью этого усовершенствования мы избегаем множественного исполнения для аргументов A_3,B_2 и A_5,B_2. Заметим, что при использовании указанного усовершенствования структура вычисления представлена ориентированным ациклическим графом, а не более привычной для рекурсивных процедур древовидной структурой.

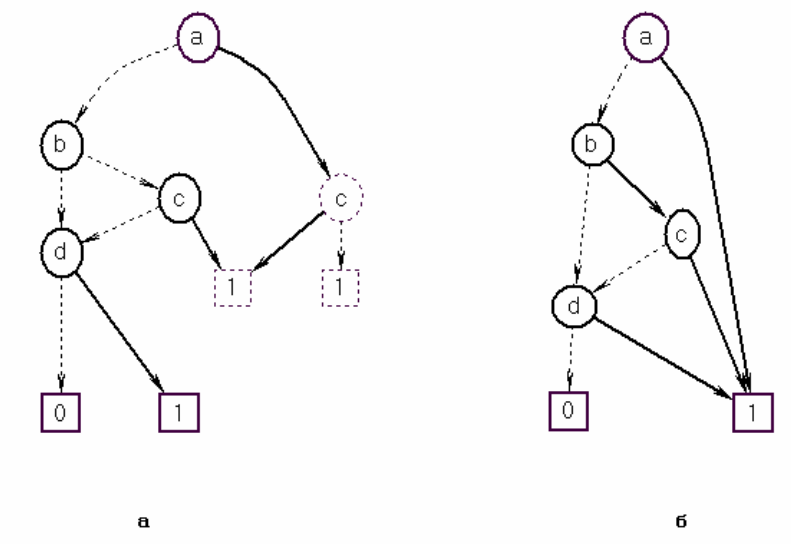


Рис. 8. Генерация результата для операции APPLY.
Структура рекурсивных вызовов по своей природе определяет неприведенный граф (слева). С помощью применения правил сокращения в конце каждого рекурсивного вызова сразу же строится приведенный граф (справа)

Каждый шаг вычисления возвращает в качестве результата вершину генерируемого графа. Структура рекурсивной оценки по своей природе определяет неприведенный граф, в котором каждый шаг вычисления задает вершину, помеченную расщепляющей переменной и имеющую в качестве потомков результаты рекурсивных вызовов. Такой граф для нашего примера показан на рис. 8. Чтобы сразу же генерировать приведенный граф, на каждом шаге выполнения мы пытаемся «уклониться» от создания новой вершины с помощью применения тестов, соответствующих описанным ранее правилам преобразования. Предположим, что шаг выполнения содержит расщепляющую переменную x, и рекурсивные вычисления возвращают вершины v_0 и v_1. Вначале мы проверяем, выполняется ли условие $v_0 = v_1$, и, если выполняется, то мы возвращаем эту вершину как результат процедуры. Затем мы проверяем, не содержит ли уже сгенерированный граф какую-нибудь вершину v, такую, что ПЕР(v) = x, ЛЕВ(v) = v_0 и ПРАВ(v) = v_1. Эта проверка сопровождается поддержкой хэш-таблицы, содержащей запись для каждой нетерминальной вершины v в сгенерированном файле с ключом

<ПЕР(v), ПРАВ(v), ЛЕВ(v)>.

Если искомая вершина найдена, она возвращается как результат процедуры. В противном случае вершина добавляется в граф, ее запись добавляется в хэш-таблицу, и вершина возвращается как результат процедуры. Подобным же образом в хэш-таблице оказываются терминальные вершины, их ключами являются метки. Новая терминальная вершина генерируется только в том случае, если не имеется вершины с нужной меткой. В нашем примере этот процесс не создает вершины, изображенные пунктиром в левой части рис. 8. Напротив, граф в правой части рис. 8 генерируется непосредственно. Заметим, что этот граф представляет функцию

$$a + b*c + d,$$

которая и является результатом применения операции OR к двум функциям-аргументам.

Использование таблицы для устранения множественного исполнения данной пары вершин ограничивает сложность операции APPLY и одновременно задает границу размера результата. А именно, пусть функции f и g представлены OBDD-диаграммами с количеством вершин m_f и m_g, соответственно. Тогда может быть не более чем $m_f m_g$ уникальных аргументов для вычисления, и каждое вычисление добавляет не более одной вершины к генерируемому результату. Имея хорошую реализацию хэш-таблицы, мы можем провести каждый шаг вычисления в среднем за константное время. Таким образом, и сложность алгоритма, и размер генерируемого результирующего графа должны оцениваться как $O(m_f m_g)$.

2.2. Операция RESTRICT

Вычисление рестрикции функции, представленной любым видом бинарной диаграммы решений, довольно просто. Чтобы выполнить рестрикцию переменной x до значения k, мы перенаправим любую дугу, ведущую к вершине v, для которой ПЕР(v) = x, либо в ЛЕВ(v) для k = 0, либо в ПРАВ(v) для k=1. На рис. 9 представлена рестрикция переменной b в функции

$$b*c + a*\overline{b} *\overline{c}$$

до значения 1. Исходная функция задана OBDD в левой части. После перенаправления дуг вершина b исключается из обхода, что показано в центральной части.

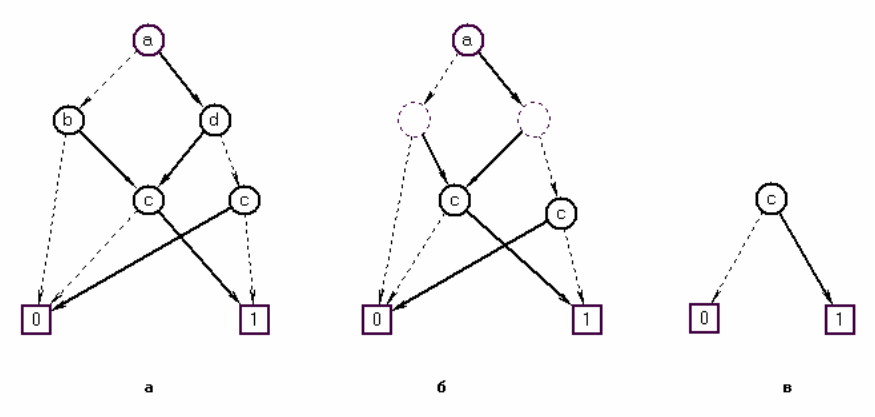


Рис. 9. Пример операции RESTRICT.
Рестрикция переменной b (слева) до значения 1 приводит к исключению из обхода вершин, помеченных b (в центре) и к получению приведенного графа (справа)

Как видно из этого примера, прямая реализация техники рестрикции может дать в итоге неприведенный граф. Взамен этого вначале производится обход исходного графа в глубину. Каждый рекурсивный вызов имеет своим аргументом вершину в исходном графе и возвращает в виде результата вершину генерируемого графа. Чтобы гарантировать, что генерируемый граф является приведенным, процедура поддерживает хэш-таблицу, содержащую запись для каждой вершины генерируемого графа, применяя те же правила сокращения, что и при операции APPLY. В нашем примере результат может быть OBDD-представлением функции *c*, как показано в правой части рис. 9.

Вычисление рестрикции функции *f*, OBDD-представление которой имеет m_f вершин, содержит не более чем m_f рекурсивных вызовов, каждый из которых генерирует не более одной вершины результирующего графа. Имея хорошую реализацию хэш-таблицы, каждый шаг рекурсии требует в среднем константного времени. Таким образом, и сложность алгоритма, и размер генерируемого результирующего графа можно оценить как $O(m_f)$.

2.3. Производные операции

Как было описано выше, значительное количество операций над булевыми функциями может быть выражено в терминах алгебраических опера-

ций и операции рестрикции. Алгоритмы APPLY и RESTRICT дают способ реализации этих операций. Далее, для каждой из этих операций ее сложность и размер генерируемого графа ограничены некоторой полиномиальной функцией относительно функций-аргументов. Обозначим размер OBDD-представления функции f как m_f. Пусть даны функции f и g и «состояния безразличности», выраженные функцией d, тогда мы можем вычислить эквивалентность f и g для «небезразличных» состояний за время $O(m_f m_g m_d)$. Мы можем вычислить композицию функций f и g с двумя рестрикциями и тремя вызовами APPLY. Этот подход дает временную сложность $O(m_f^2 m_g^2)$. Реализуя все вычисление за один проход, можно сократить эту оценку до $O(m_f m_g^2)$. Наконец, можно провести квантификацию переменной функции f за время $O(m_f^2)$.

2.4. Характеристики производительности

Использование OBDD для решения проблемы требует выражения задания в виде серии операций на булевых функциях, подобных описанным выше. Как мы видели, все эти операции могут быть реализованы с помощью алгоритмов, полиномиальных по сложности относительно размера диаграмм, представляющих аргументы. В результате основанные на OBDD символьные булевы операции имеют два преимущества по сравнению с другими общепринятыми подходами.

Во-первых, до тех пор, пока размеры графов остаются приемлемыми, все вычисление в целом оказывается контролируемым.

Во-вторых, хотя размеры графов могут возрастать с каждой успешной операцией, любая отдельная операция имеет разумную производительность в худшем случае.

Напротив, многие другие представления булевых функций не обладают этим свойством «постепенного снижения производительности». Например, справедливо следующее свойство.

Свойство Г. *Даже если функция имеет умеренно компактное представление суммы произведений, ее дополнение может быть экспоненциального размера.*

2.5. Технология реализации

С точки зрения реализации, основанные на OBDD символьные операции имеют характеристики, отличающие их от многих других вычислительных задач. Во время вычисления конструируются и уничтожаются тысячи гра-

фов, содержащие тысячи вершин каждый. Информация представлена при использовании OBDD-представлений скорее самой их структурой, чем ассоциированными с ними данными, и в силу этого к каждой отдельной вершине прилагаются очень небольшие вычислительные действия. Таким образом, вычисление носит высоко динамичный характер, не позволяющий делать предварительных предположений об обращениях к памяти. В настоящее время наиболее успешные варианты реализации были получены на рабочих станциях с большой физической памятью, на которых уделялось много внимания программированию процедур управления памятью.

Чтобы достичь максимальной производительности, было бы желательно использовать потенциал конвейерных и параллельных компьютеров. В задачах символьного анализа параллелизм может существовать на *макроуровне*, когда многие операции могут быть исполнены одновременно, и на *микроуровне*, когда многие вершины внутри заданного OBDD-представления могут быть обработаны одновременно. В сравнении с другими задачами, которые были успешно реализованы на векторных и параллельных компьютерах, обработка с использованием OBDD требует заметно большей коммуникации и синхронизации между вычисляющими элементами и заметно меньших локальных вычислений. Эта задача пока остается вызовом для разработчиков архитектур параллельных компьютеров, программных моделей и языков программирования. Тем не менее, некоторые из первых попыток оказались многообещающими. Исследователи успешно использовали векторную обработку и показали хорошие результаты при исполнении на мультипроцессорах с разделенной памятью. Оба подхода используют параллелизм на микроуровне с помощью реализации операции APPLY в виде обхода графов-аргументов в ширину, в противоположность общепринятому для большинства других реализаций обходу в глубину.

3. ПРЕДСТАВЛЕНИЕ МАТЕМАТИЧЕСКИХ ОБЪЕКТОВ

Некоторые приложения, чаще всего в области разработки цифровых систем связи, требуют прямой реализации и обработки булевых функций. Однако в общем случае мощность символьной булевой обработки состоит в способности бинарных значений и булевых операций представлять и воплощать разнообразные области математики.

Таблица 3

**Примеры систем,
которые могут быть представлены булевыми функциями**

Класс	**Типичные операции**	**Типичные тесты**
Логика	$\wedge$, $\vee$, $\neg$, $\forall$, $\exists$	выполнимость, импликация
Конечные области	*Проблемно-зависимые*	эквивалентность
Функции	Аппликация, композиция	эквивалентность
Множества	$\cup$, $\cap$, –	подмножество
Отношения	композиция, замыкание	симметричность, транзитивность

В табл. 3 приводятся примеры некоторых разделов математики, в которых объекты могут быть представлены, обработаны и проанализированы с помощью символьных булевых операций, так как лежащие в их основе области являются конечными. Предоставляя унифицированную структуру для различных математических систем, символьная булева обработка дает возможность решать не только проблемы в отдельных областях, но и проблемы, затрагивающие различные области математики. Например, разработаны программы для анализа последовательного поведения цифровых схем, которые включают операции над всеми перечисленными в табл. 3 областями. Требуемые свойства системы выражаются в виде логических формул. Поведение системы задается функциями следующего состояния схемы. Анализатор вычисляет множества состояний, имеющих определенные свойства. Структура переходов в системе конечного автомата представлена в виде отношения. Во время исполнения анализатор может быстро перемещаться между представлениями, используя только OBDD как базисную структуру данных. Более того, свойство каноничности OBDD позволяет с легкостью распознавать такие условия, как сходимость, и определять, существует ли решение заданной проблемы.

Ключом к использованию возможностей символьной булевой обработки является выражение исходной проблемы в форме, в которой все объекты представлены булевыми функциями. В оставшейся части данной главы мы опишем некоторые стандартные техники, которые были разработаны для этой цели. Имея достаточный опыт и практику, можно представить в таком виде удивительно широкий класс проблем. Математические основы для

этого представления были разработаны уже давно. Ни одна из рассматриваемых техник не опирается на особенности OBDD-представления — они могут быть реализованы для любого из множества представлений. Разработка OBDD-представления просто расширила спектр проблем, которые могут быть разрешены практически. Однако тем самым она усилила мотивацию к представлению различных проблем в терминах символьных булевых операций.

3.1. Кодирование конечных областей

Рассмотрим конечное множество элементов A, $|A| = N$. Мы можем кодировать элемент из A с помощью вектора из n бинарных значений, где $n = \lceil \log_2 N \rceil$. Это кодирование обозначается функцией $\sigma: A \to \{0, 1\}^n$, которая сопоставляет элементам из A различные битовые n-вектора. Пусть $\sigma_i(a)$ обозначает i-й бит этого вектора. Функция $f: A \to A$, отображающая элементы A на элементы A, представлена вектором из n булевых функций $\vec{f}$, где каждая функция f_i: $\{0, 1\}^n \to \{0, 1\}$ определяется как

$$f_i(\sigma(a)) = \sigma_i(f(a)).$$

Во многих реальных приложениях области обладают «естественным» порядком (пример — бинарное кодирование целых чисел); для других областей порядок строится искусственно.

Например, символьный симулятор COSMOS использует OBDD для символьного расчета поведения транзисторной схемы. Такой симулятор может быть использован для автоматической генерации тестов для поиска ошибок в схеме и для формальной проверки того, отвечает ли поведение схемы некоторым заданным критериям. В модели схемы напряжение в узлах представлено трехзначным сигнальным набором, в котором значения 0 и 1 обозначают низкое и высокое напряжение, а третье значение, X, обозначает неизвестное или потенциально нецифровое напряжение. Во время символьного моделирования состояния вершин должны быть вычислены как трехзначные функции над множеством булевых переменных, введенных пользователем для представления первичных входных данных или начального состояния. COSMOS представляет состояние вершины парой OBDD-диаграмм. А именно, он кодирует каждый из $N = 3$ элементов сигнального набора вектором из $n = 2$ бинарных значений в соответствии с кодирующей функцией:

$$\sigma(0) = [0, 1], \sigma(1) = [1, 0], \sigma(X) = [1, 1].$$

Функции следующего состояния, вычисляемые симулятором, полностью определяются в соответствии с этим булевым кодированием, что дает возможность точно описать булевыми функциями поведение схемы. В табл. 4 приведен пример трехзначного расширения операций AND, OR и NOT. Заметим, что операции дают значение X в каждом случае, когда неизвестный аргумент может вызвать неопределенность в значении функции. Пусть $[a_1, a_0]$ обозначает кодировку трехзначного сигнала a, тогда трехзначная операция может быть полностью выражена в терминах булевых операций:

$$[a_1, a_0] \cdot_t [b_1, b_0] = [a_1 \cdot b_1, a_0 + b_0],$$
$$[a_1, a_0] +_t [b_1, b_0] = [a_1 + b_1, a_0 \cdot b_0],$$
$$\overline{[a_1, a_0]}^t = [a_0, a_1].$$

Во время функционирования симулятор действует подобно обычному логическому симулятору, управляемому событиями. Вначале каждый внутренний узел инициализируется значением [1, 1], означающим, что значение узла неизвестно независимо от обстоятельств. В процессе моделирования состояния узлов модифицируются с помощью вычисления булевого представления функции следующего состояния с применением операции APPLY. Каждый раз, когда состояние узла перевычисляется, старое состояние сравнивается с новым, и в случае, когда они неэквивалентны, создается событие для каждого разветвления этого узла. Процесс продолжается до тех пор, пока список событий не становится пустым, указывая на то, что сеть находится в стабильном состоянии. Этот метод обработки событий требует наличия эффективного способа проверки эквивалентности.

Таблица 4

Трехзначное расширение операций AND, OR и NOT. Третье значение X обозначает неизвестное или потенциально нецифровое напряжение

$\cdot_t$	0	1	X
0	0	0	0
1	0	1	X
X	0	X	X

$+_t$	0	1	X
0	0	1	X
1	1	1	1
X	X	1	X

a	$\overline{a}^t$
0	1
1	0
X	X

3.2. Множества

Пусть задано базисное множество A. Мы можем представлять и обрабатывать его подмножества, используя «характеристические функции». Множество $S \subseteq A$ обозначается функцией χ_S: $\{0, 1\}^n \to \{0, 1\}$,

$$\chi_S(\vec{x}) = \sum_{a \in S} \prod_{1 \le i \le n} x_i \,\overline{\otimes}\, \sigma_i(a),$$

где $\overline{\oplus}$ представляет собой дополнение операции EXCLUSIVE-OR. Теперь операции над множествами (подмножествами) могут быть реализованы с помощью булевых операций на их характеристических функциях, например:

$$\chi_\varnothing := \mathbf{0},$$
$$\chi_{S \cup T} := \chi_S + \chi_T,$$
$$\chi_{S \cap T} := \chi_S \cdot \chi_T,$$
$$\chi_{S-T} := \chi_S \cdot \overline{\chi_T}.$$

Множество S является подмножеством T в том и только том случае, если $\chi_S \cdot \overline{\chi_T} = \mathbf{0}$. Многие приложения, использующие OBDD, строят и обрабатывают множества даже без явной нумерации их элементов. Вместо этого можно представить (непустое) множество как множество возможных выходных значений вектора функций, т.е. можно полагать, что функция $\vec{f}$ обозначает множество

$$\{a \mid \sigma(a) = \vec{f}(\vec{b}) \text{ для некоторого } \vec{b} \in \{0,1\}^n\}.$$

Это представление может оказаться удобным для приложений, в которых анализируемая система представлена вектором функций. Модифицируя эти функции, мы можем также представлять подмножества состояний системы.

3.3. Отношения

Мы можем определить k-арное отношение как множество упорядоченных кортежей размерности k. Следовательно, отношения также могут быть представлены и обработаны с помощью характеристических функций. Рассмотрим в качестве примера бинарное отношение $R \subseteq A \times A$, обозначаемое булевой функцией χ_R, которая определяется следующим образом:

$$\chi_R(\vec{x}, \vec{y}) = \sum_{a \in A} \sum_{\substack{b \in A \\ aRb}} \left[\prod_{1 \le i \le n} x_i \,\overline{\oplus}\, \sigma_i(a) \right] \left[\prod_{1 \le i \le n} y_i \,\overline{\oplus}\, \sigma_i(b) \right].$$

С помощью этого представления мы можем производить такие действия, как пересечение, объединение и разность отношений, применяя булевы операции к их характеристическим функциям.

Комбинируя функциональную композицию и квантификацию переменных, мы можем также получать составные отношения:

$$\chi_{R^\circ S} := \exists\ \vec{x}\ [\chi_R(\vec{x},\vec{z}) \cdot \chi_S(\vec{z},\vec{y})],$$

где $R \circ S$ обозначает композицию отношений R и S. Квантификация над вектором переменных включает квантификацию каждого элемента вектора в любом порядке. Далее, мы можем вычислить транзитивное замыкание отношения, используя метод неподвижной точки. Функция χ_{R^*} вычисляется как предел последовательности функций χ_{Ri}, каждую из которых определяет отношение

$$R_0 = I,$$
$$R_{i+1} = I \cup R \circ R_i,$$

где I обозначает тождественное отношение. Процесс вычисления сходится, когда он достигает такой итерации i, что $\chi_{Ri} = \chi_{Ri+1}$, для чего вновь потребуется эффективная проверка эквивалентности. Если мы предположим, что R представляет граф, с вершиной для каждого элемента A и дугой для каждого элемента R, тогда отношение R_i обозначает пары вершин, достижимые по путям длиной не более чем i дуг. Следовательно, вычисление должно сходиться максимум за N-1 итераций, где $N = |A|$. Техника, известная как «метод итеративных квадратур», сокращает максимальное количество итераций до $n = \lceil \log_2 N \rceil$. Каждая итерация вычисляет отношение $R_{(i)}$, указывающее на пары вершин, достижимых по путям не длиннее 2^n:

$$R_{(0)} = I \cup R,$$
$$R_{(i+1)} = R_{(i)} \circ R_{(i)}.$$

Многие приложения OBDD содержат обработку отношений над очень большими множествами, и поэтому сокращение числа итераций с N (например, 10^9) до n (например, 30) может быть очень существенным.

4. ДРУГИЕ ПРИМЕНЕНИЯ

OBDD широко используются в разработке, верификации и тестировании цифровых систем связи. Ниже будут описаны методы их применения в некоторых областях.

4.1. Верификация

OBDD могут быть использованы непосредственно для установления эквивалентности двух комбинаторно-логических схем. Эта проблема возникает в случаях, когда необходимо сравнить схему и сеть, полученную из спецификаций системы, либо когда требуется проверить, не изменил ли логический оптимизатор функциональность схемы. Вначале с помощью операции APPLY вычисляются функциональные представления обеих сетей, и затем проверяется их эквивалентность. Этот метод позволяет также сравнивать две последовательные системы, если они используют одно и то же кодирование состояний, а именно — если они имеют идентичные выходные данные и функции перехода (к следующему состоянию).

4.2. Коррекция ошибок схемы

Не довольствуясь простым обнаружением ошибок в логической схеме, можно создавать методы автоматической коррекции дефектных схем. Существующие методы коррекции включают рассмотрение относительно небольшого класса потенциальных ошибок, таких, как одиночный некорректный логический элемент, и определение того, может ли какой-либо вариант данной цепи достичь требуемой функциональности. Этот анализ демонстрирует мощь операций квантификации при вычислении проекций, в данном случае производя проекцию первичных входных значений посредством универсальной квантификации.

Такой анализ может быть проведен символически, с помощью кодирования возможных функций элементов булевыми переменными. Как показано на примере (рис. 10), произвольный *k*-входовой канал может быть моделирован 2^k-входовым мультиплексором, где операция элемента определена входными данными мультиплексора $\vec{a}$. Рассмотрим контур N с одним входом, в котором один из логических элементов заменен таким блоком, это дает в итоге функциональность сети $N(\vec{x},\vec{a})$, где $\vec{x}$ представляет множество первичных входных значений. Предположим, что желаемая функциональность есть $S(\vec{x})$. Наша задача — определить, могут ли (и если могут, то каким образом) две функции быть сделаны идентичными для всех значений первичных данных посредством подходящего «программирования» элемента. Это включает вычисление функции $C(\vec{a})$, определенной следующим образом:

$$C(\vec{a}) = \forall \vec{x} \ \ [N(\vec{x},\vec{a}) \ \overline{\oplus} \ S(\vec{x})].$$

Хотя серьезные ошибки схемы не могут быть исправлены таким образом, он позволяет избежать утомительной коррекции часто встречающихся ошибок, таких, как неправильное расположение инверторов или использование некорректного типа логического элемента. Эта задача также оказывается полезной в логическом синтезе, когда требуется изменить схему таким образом, чтобы она соответствовала измененной спецификации.

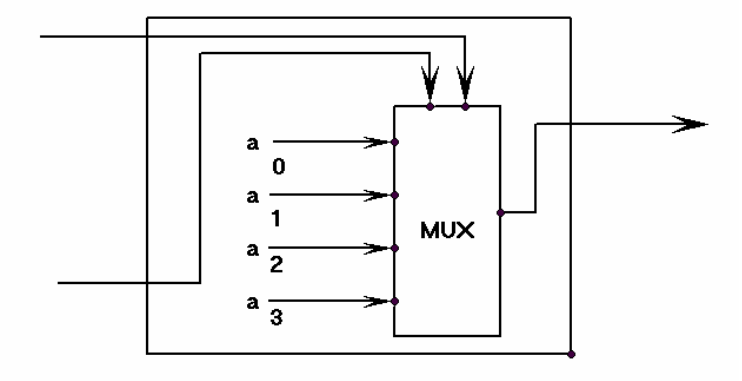


Рис. 10. Блок функции всеобщности.
Присваивая различные значения переменным $\vec{a}$,
можно реализовать произвольную 2-входовую операцию

4.3. Анализ чувствительности

Второй класс приложений включает составление характеристик того влияния, который оказывают изменения значений сигналов различных проводов комбинаторной схемы. А именно, для каждого значения сигнала *s* мы хотим вычислить булеву разницу для каждого первичного выходного значения относительно *s*. Этот анализ может быть проведен символически, с помощью введения в сеть «преобразователей сигнального провода», как показано на рис. 11. Для каждого провода, в обычной ситуации передающего сигнал *s*, мы избирательно изменяем значение сигнала, которое теперь равно *s'*, при помощи булевого значения P_s: $s' = s \oplus P_s$.

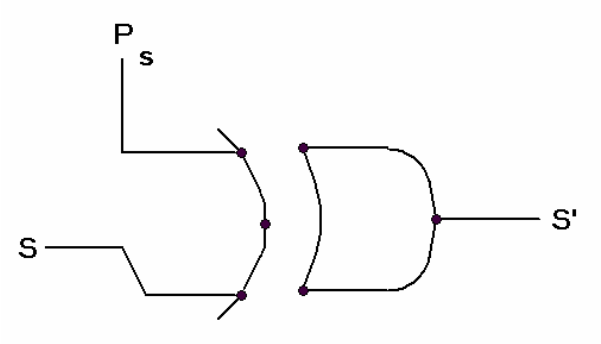


Рис. 11. Преобразователь сигнального провода.
Ненулевое значение P_s изменяет значение, передаваемое по этому проводу

Мы можем определить условия, при которых некоторое выходное значение схемы оказывается чувствительным к значению сигнального провода, сравнивая выходные значения исходной и измененной схем, как показано на рис. 12. Как видно из этого рисунка, мы можем даже вычислить влияние каждого однопроводного преобразования в схеме за одно символическое вычисление. А именно, пронумеруем каждый сигнальный провод числами от 1 до m-1 и введем множество $\lceil \log m \rceil$ «перестановочных переменных» $\vec{r}$. Каждый перестановочный сигнал P_s затем определяется как функция, принимающая значение 1 в случае, когда перестановочные переменные являются бинарным представлением номера, присвоенного сигналу s. В терминах построения логических схем это эквивалентно генерации перестановочных сигналов декодером с $\vec{r}$ на входе. Полученная функция $T(\vec{x}, \vec{r})$ принимает значение 1, если исходная схема и схема, полученная с помощью перестановки $\vec{r}$, выдают одни и те же выходные значения на входных значениях $\vec{x}$.

Одной областью приложения такого анализа чувствительности является автоматическая генерация тестов. Функция чувствительности описывает множество всех тестов для каждой одиночной ошибки. Предположим, что сигнальный провод, помеченный бинарным номером $\vec{b}$, в обычной ситуации имеет в схеме функцию $s(\vec{x})$. Тогда входной шаблон $\vec{a}$ будет обнаруживать ошибку типа «отсутствие сигнала на проводе 1» в том и только том случае, если $\overline{T(\vec{a},\vec{b})} \bullet \overline{s(\vec{a})} = 0$. Подобным же образом, шаблон $\vec{a}$ будет

обнаруживать ошибку типа «отсутствие сигнала на проводе 0» в том и только том случае, если

$$\overline{T(\vec{a},\vec{b})} \bullet \overline{s(\vec{a})} = 1.$$

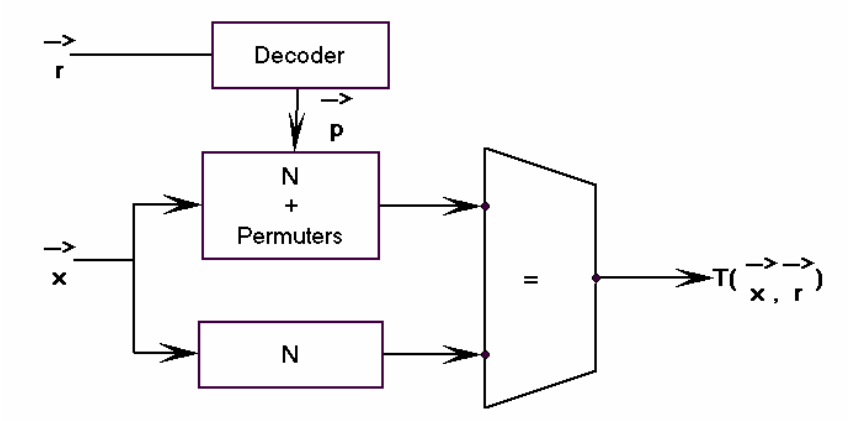


Рис. 12. Вычисление чувствительных к однопроводным преобразованиям участков. Каждое присваивание переменной $\vec{r}$ вызывает изменение значения ровно одного провода

Этот метод также может быть распространен на последовательные схемы и на схемы, представленные на уровне переключателей.

Второй областью приложения является комбинационно-логическая оптимизация. Для сигнального провода, имеющего бинарный номер $\vec{b}$, функция $T(\vec{x},\vec{y})$ представляет набор «безразличностей» для каждого провода схемы, то есть таких случаев, когда выходные значения схемы не зависят от сигнального значения этого провода. Руководствуясь этой информацией, оптимизатор схемы может применять такие преобразования, как удаление сигнального провода или перемещение провода на другой выходной элемент. Недостаток этого подхода состоит в том, что функция чувствительности должна вычисляться заново каждый раз, как оптимизатор изменяет схему. Альтернативный подход дает более ограниченный, но и более компактный набор функций «безразличностей», в котором наборы «безразличностей» остаются действительными даже в случаях, когда меняется структура схемы.

4.4. Вероятностный анализ

Существует метод для статистического анализа влияния меняющейся задержки сигнала в цифровой схеме. Это приложение OBDD выглядит особенно интригующим, поскольку здравый смысл заставляет считать, что такой анализ требует вычисления реальных вариаций параметров и в силу этого не может быть закодирован булевыми переменными.

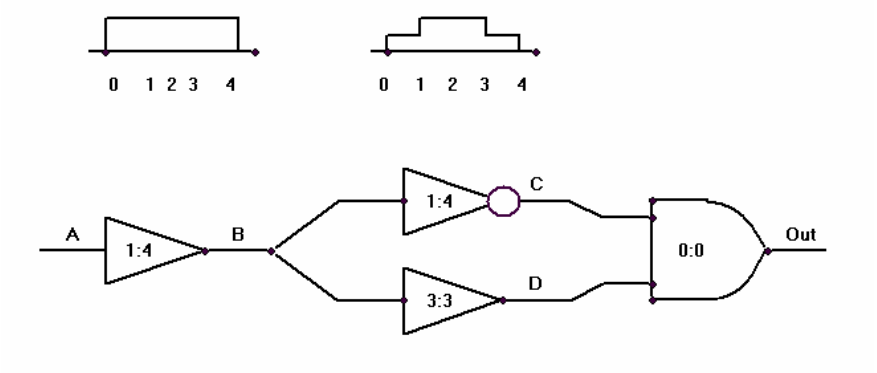


Рис. 13. Схема с неопределенными задержками.
Логические элементы помечены своими минимальными и максимальными задержками.
Инверторы имеют распределение задержек

Рассмотрим сеть логических элементов, в которой каждый элемент имеет задержку, задаваемую некоторым вероятностным распределением. Эта схема демонстрирует диапазон вариантов поведения, некоторые из которых классифицируются как нежелательные. «Выход» определяется как вероятность того, что нежелательное поведение не встретится. На рис. 13 приведен пример простой схемы, где два из логических элементов имеют переменное распределение задержек; мы хотим оценить вероятность появления сбоя в вершине Out, в то время как входной сигнал A совершает переход из 0 в 1. Нужно провести анализ ситуации, когда сигнал A изменяет свое значение на 1 в момент времени 0. Сигналы C и D совершают переход, для которого известно вероятностное распределение времени перехода. Необходимо провести однократный простой анализ для расчета вероятной формы волны для всех сигналов, как если бы они распределялись независимо. Затем мы сможем легко вычислить поведение каждого выходного значения схемы в соответствии с функцией элемента и формой входной волны. Например, если мы рассматриваем сигналы C и D как независимые, мы можем

вычислить вероятность приближающегося перехода в вершине Out в момент времени t как произведение (1) вероятности того, что C совершает переход в момент t и (2) вероятности того, что D не совершает перехода во время ≤ t. Это приводит к вероятностному распределению перехода, помеченному «Out (Independent)». Конечная вероятность появления перехода (то есть сбоя) оказывается равной 30%. В действительности, разумеется, время перехода сигналов C и D в значительной степени коррелирует друг с другом — оба подвержены влиянию задержки начального буферного элемента. В силу этого более тщательный анализ даст вероятностное распределение времени перехода, помеченное «Out (Actual)», в результате конечная вероятность появления перехода оказывается равной 12.5%. Таким образом, упрощенный анализ недооценивает выход схемы. В других случаях упрощенный анализ может переоценивать выход схемы.

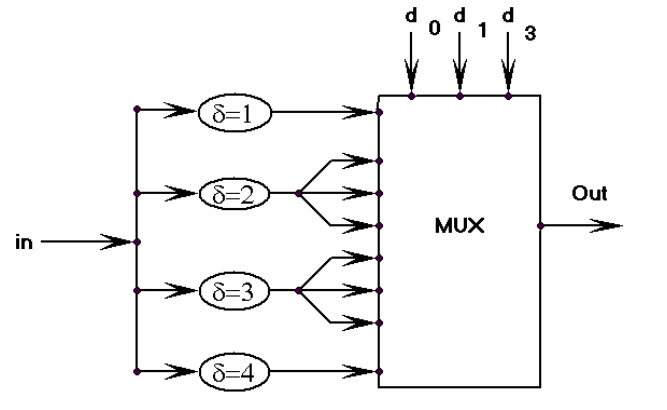


Рис. 14. Моделирование неопределенных задержек.
Булевы переменные управляют выбором задержки.
Сигналы повторяются в соответствии с распределением задержек

Чтобы решить эту проблему с помощью символьного булева анализа, мы должны принять два ограничения. Во-первых, все задержки схемы должны иметь целочисленные значения (с помощью подходящего выбора единицы времени), в силу чего переходы будут совершаться в дискретных временных точках. Во-вторых, вероятность задержки должна быть кратной значению $1/k$, где k — некоторая степень двойки. Например, оба переменных элемента на рис. 13 имеют задержки от 1 до 4. Один имеет равномерное распределение задержек [1/4, 1/4, 1/4, 1/4], тогда как у другого распре-

деление задержек приближается к нормальному [1/8, 3/8, 3/8, 1/8]. Затем значение задержки для элемента может быть закодировано множеством из log k булевых переменных, как показано на рис. 14. А именно, мы моделируем компонент схемы k-входовым мультиплексором, в котором значение задержки, имеющее вероятность c/k, подается на с входов. Затем схема рассчитывается с использованием символьного расширения обычного алгоритма симуляции логических схем. Значение сигнала в вершине N в каждый момент времени t является булевой функцией $N(t)$ от переменных задержки.

Таблица 4

Условия задержки для схемы на рис. 14

$A \rightarrow B$

Задержка	*Условия*
1	$\overline{e_1} \cdot \overline{e_0}$
2	$\overline{e_1} \cdot e_0$
3	$e_1 \cdot \overline{e_0}$
4	$e_1 \cdot e_0$

$B \rightarrow C$

Задержка	*Условие*
1	$\overline{d_2} \cdot \overline{d_1} \cdot \overline{d_0} \cdot$
2	$\overline{d_2} \cdot (d_1 + d_0)$
3	$d_2 \cdot (\overline{d_1} + \overline{d_0})$
4	$d_2 \cdot d_1 \cdot d_0$

Предположим, что в примере на рис. 15 переменные $[e_1, e_0]$ кодируют задержку между A и B, тогда как переменные $[d_2, d_1, d_0]$ кодируют задержку между B и C, как показано в табл. 4. Для времени $t < 0$ функции вершин будут следующими: $A(t) = B(t) = D(t) = Out(t) = \mathbf{0}$, $C(t) = \mathbf{1}$. Для времени $t \geq 0$ вершина A имеет функцию $A(t) = \mathbf{1}$, остальные функции вычисляются следующим образом:

$$\begin{aligned} B(t) = {} & \overline{e_1} \cdot \overline{e_0} \cdot A(t\text{-}1) \\ & + \overline{e_1} \cdot e_0 \cdot A(t\text{-}2) \\ & + e_1 \cdot \overline{e_0} \cdot A(t\text{-}3) \\ & + e_1 \cdot e_0 \cdot A(t\text{-}4), \end{aligned}$$

$$C(t) = \overline{d_2} \cdot \overline{d_1} \cdot \overline{d_0} \cdot \overline{B(t-1)}$$
$$+ \overline{d_2} \cdot (d_1 + d_0) \cdot \overline{B(t-2)}$$
$$+ d_2 \cdot (\overline{d_1} + \overline{d_0}) \cdot \overline{B(t-3)}$$
$$+ d_2 \cdot d_1 \cdot d_0 \cdot \overline{B(t-4)},$$
$$D(t) = B(t - 3),$$
$$Out(t) = C(t) \cdot D(t).$$

Значение выходного сигнала мы можем получить из этих уравнений как Out(t) = **0** для t ≤ 3 и t ≥ 8. Для других значений времени значение сигнала будет вычисляться следующим образом:

$$Out(4) = d_2 \cdot d_1 \cdot d_0 \cdot \overline{e_1} \cdot \overline{e_0},$$
$$Out(5) = d_2 \cdot d_1 \cdot d_0 \cdot \overline{e_1} \cdot e_0,$$
$$Out(6) = d_2 \cdot d_1 \cdot d_0 \cdot e_1 \cdot \overline{e_0},$$
$$Out(7) = d_2 \cdot d_1 \cdot d_0 \cdot e_1 \cdot e_0.$$

Мы можем вычислить булеву функцию, показывающую условия задержки, при которых встречается некоторое нежелательное поведение. Например, мы можем вычислить вероятность того, что сбой происходит в вершине *Out*, как $G = \Sigma Out(t)$. В этом случае мы можем вычислить $G = d_2 \cdot d_1 \cdot d_0$, то есть вероятность того, что сбой происходит в том и только том случае, когда задержка между B и C равна 4.

Пусть дана булева функция, представляющая условия, при которых происходит некоторое событие. Мы можем вычислить вероятность события путем вычисления плотности функции, то есть доли присваиваний переменным, на которых функция принимает значение 1. С помощью Шенноновского раскрытия мы можем задать плотность $\rho(f)$ функции f рекурсивным образом:

$$\rho(\mathbf{1}) = 1$$
$$\rho(\mathbf{0}) = 0$$
$$\rho(f) = 1/2\ [\rho(f|_{x\leftarrow 0}) + \rho(f|_{x\leftarrow 1})].$$

Таким образом, при наличии OBDD-представления f мы можем вычислить плотность за линейное время путем обхода графа в глубину, помечая каждую вершину плотностью функции, обозначающей ее подграф. На

рис. 15 изображено такое вычисление для OBDD, представляющего условия, при которых вершина C на рис. 14 содержит переход в момент времени 6, помечая ее вероятностью этого события, равной 7/32.

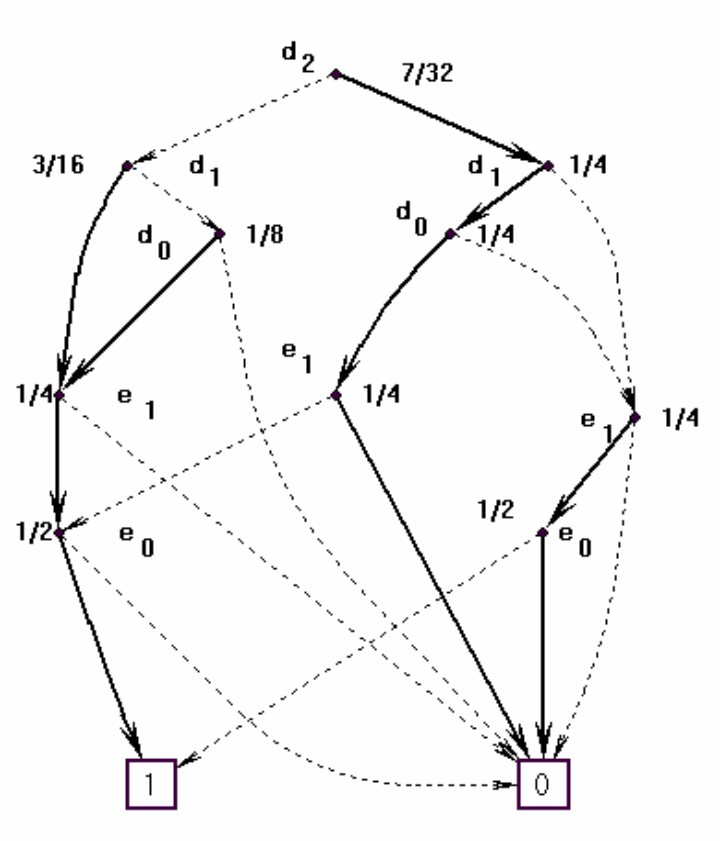


Рис. 15. Вычисление плотности функции.
Каждая вершина помечена долей присваиваний переменным, дающих в результате значение 1

Как демонстрирует это приложение, основанный на OBDD символьный анализ может быть применен к системам со сложными вариациями параметров. Хотя при этом требуется упростить проблему таким образом, чтобы было возможно рассматривать только дискретные вариации, все-таки могут быть получены хорошие результаты. Основным достоинством этого подхо-

да по сравнению с другими упрощенными методами вероятностного анализа (например, измерение управляемости и наблюдаемости) является то, что он точно вычисляет влияние корреляций между стохастическими значениями.

4.5. Анализ конечных автоматов

Многие задачи в таких областях, как верификация цифровых систем, проверка протоколов и оптимизация последовательных систем, требуют детальной характеризации конечного автомата над последовательностью переходов. Классические алгоритмы для задачи характеризации осуществляют явное представление автомата в виде диаграммы переходов и затем анализируют структуру его путей и циклов. Для этого применения также были разработаны более специальные методы, использующие характеристики верифицируемой системы, например, того, что схема является синхронной и детерминированной и что спецификация требует анализировать только ограниченное количество тактов. Однако эти методы непрактичны, если количество состояний становится большим. К сожалению, даже относительно маленькие цифровые системы могут иметь очень большие пространства состояний. К примеру, один 32-битный регистр может иметь около $4 * 10^9$ состояний.

Существуют «символьные» методы описания автоматов, в которых структура переходов представлена в виде булевой функции. Вначале выбираются бинарное кодирование состояний системы и входной алфавит. Поведение, связанное со следующим состоянием, описывается как отношение, задаваемое характеристической функцией $\delta(\vec{x},\vec{o},\vec{n})$, которая принимает значение 1 в случае, когда входное значение $\vec{x}$ может вызвать переход из состояния $\vec{o}$ в состояние $\vec{n}$.

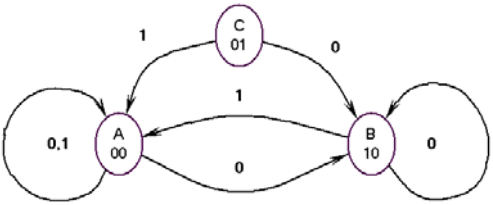


Рис. 16. Явное представление недетерминированного конечного автомата. Размер представления растет линейно относительно количества состояний

На рис. 17 приведен пример OBDD-представления недетерминированного автомата, диаграмма переходов которого изображена на рис. 16. В этом примере три возможных состояния представлены с помощью двух бинарных значений посредством кодирования: $\sigma(A) = [0, 0]$, $\sigma(B) = [1, 0]$, $\sigma(C) = [0, 1]$. Заметим, что неиспользованное значение [1, 1] может рассматриваться как «безразличное» для аргументов $\vec{o}$ и $\vec{n}$ функции δ. В OBDD на рис. 17 эта комбинация используется как альтернативный код состояния C для упрощения OBDD-представления.

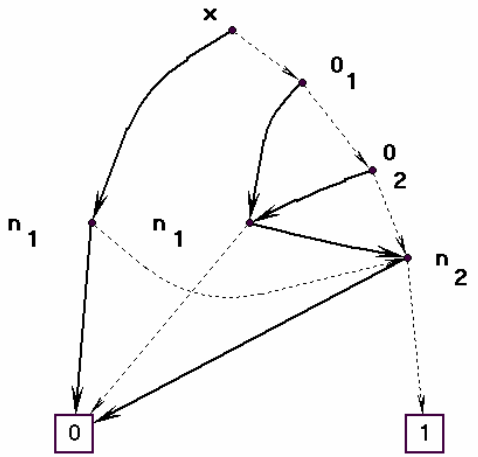


Рис. 17. Символьное представление недетерминированного конечного автомата. Количество переменных растет логарифмически относительно количества состояний

Для такого маленького автомата OBDD-представление не является улучшением по сравнению с явным представлением. Но для более сложных систем OBDD-представление может быть значительно меньшим. Основываясь на оценках сложности сетей с ограниченной шириной, описанных выше, можно охарактеризовать некоторые условия, при которых OBDD-диаграмма, кодирующая отношение переходов системы, растет линейно относительно количества компонентов системы, тогда как количество состояний растет экспоненциально. В частности, это свойство имеет место в тех случаях, когда справедливы следующие два условия:

(1) компоненты системы связаны в линейную или древесную структуру,

(2) каждый компонент поддерживает только ограниченное количество информации о состоянии других компонентов.

Как показывает пример на рис. 5, это ограничение выполняется также для кольцевых структур, так как кольцо может быть «сплющено» в линейную цепь двунаправленных ссылок.

Если имеется OBDD-представление, то свойства системы с конечным числом состояний могут быть выражены в терминах уравнений с неподвижной точкой над функцией перехода, и эти уравнения могут быть решены с использованием итеративных методов, подобных тем, которые использовались для нахождения транзитивного замыкания отношения. Рассмотрим, например, задачу определения множества состояний, достижимых из начального состояния с бинарной кодировкой $\vec{q}$ после некоторой последовательности переходов. Пусть отношение S обозначает условия, при которых для некоторого начального значения $\vec{x}$ может существовать переход из состояния $\vec{o}$ в состояние $\vec{n}$. Это отношение имеет характеристическую функцию

$$\chi_S\ (\vec{o},\vec{n}) = \exists\ \vec{x}\ [\ \delta(\vec{x},\vec{o},\vec{n})\ ].$$

Множество состояний, достижимых из состояния $\vec{q}$, имеет характеристическую функцию

$$\chi_R\ (\vec{s}\ ) = \chi_S\ (\vec{q},\vec{s}\ ).$$

С помощью этого метода были проанализированы системы с числом состояний более 10^{20}, это намного больше, чем было возможно анализировать с помощью методов, основанных на явных представлениях диаграмм переходов. Известны также усовершенствования, способствовавшие ускорению сходимости и уменьшению размера промежуточных OBDD.

К сожалению, те характеристики системы, которые гарантируют эффективность OBDD-представления отношения перехода, не дают возможности найти подходящие верхние границы для результатов, генерируемых символьным анализатором конечных автоматов. Например, существуют примеры систем, имеющих линейную структуру внутренних соединений, для которых характеристическая функция множества достижимых состояний требует OBDD-представления экспоненциального размера. С другой стороны, на практике было обнаружено, что большое количество реальных систем могут анализироваться рассмотренными методами.

Одним из приложений анализа конечных автоматов является проверка корректности последовательной цифровой схемы. Например, можно доказывать, что конечный автомат, выведенный из спецификаций системы, эквивалентен другому автомату, выведенному из схемы, даже если его кодирование состояний отличается от первого. Для этого применения также были разработаны более специальные методы, использующие характеристики верифицируемой системы, например такие, что схема является синхронной и детерминированной и что спецификация требует анализировать только ограниченное количество тактов. С их помощью, например, были верифицированы конвейерные маршруты данных, содержащие более 1000 бит состояния регистра. Такая система, имеющая более чем 10^{300} состояний, превосходит возможности современных символьных методов, основанных на диаграммах переходов.

4.6. Другие области применения

Исторически сложилось так, что вначале OBDD применялись в основном в задачах разработки цифровых систем, верификации и тестирования. Однако в последнее время они получили применение в других областях. Например, техники с использованием неподвижной точки, применяемые при символьном анализе конечных автоматов, могут быть полезными при решении множества проблем математической логики и формальных языков, если предметные области в них являются конечными. Также было обнаружено, что проблемы многих приложений могут быть сформулированы в виде набора уравнений над булевыми алгебрами, которые поддаются решению некоторым унифицированным образом.

В области искусственного интеллекта была разработана система поддержки истинности, основанная на OBDD. Она использует OBDD для представления «базы данных», то есть известных отношений между элементами. Было обнаружено, что с помощью кодирования базы данных в этой форме система может делать заключения быстрее, чем при традиционном подходе, когда она просто ведет список «известных фактов». Например, распознавание того, согласуется ли новый факт с множеством существующих фактов или следует ли он из них, представляет собой простую проверку импликации.

4.7. Возможности улучшения

Хотя существует широкое множество различных проблем, которые были успешно решены с помощью основанной на OBDD символьной обработ-

ки, имеется немало задач, для решения которых требуются более адекватные методы. Конечно, большинство этих задач являются NP-трудными и иногда даже PSPACE-трудными. Следовательно, мало вероятно, что для их решения может быть найден метод с полиномиальной сложностью. Лучшее, что можно сделать, — это разработать методы, дающие приемлемую производительность для большинства интересующих нас задач.

Одна из возможностей на пути к этому состоит в улучшении самого представления. При работе с цифровыми системами, содержащими умножители и другие функции, включающие сложные отношения между управляющими сигналами и сигналами данных, OBDD-представления быстро становятся нереально большого размера. Было предложено несколько методов, основанных на тех же общих принципах, что и OBDD, но с менее строгими ограничениями на структуры данных. Среди них IFE-представления (или «If-Then-Else дэги»), где условие проверки в каждой вершине может быть более сложной функцией, чем обычная проверка переменной, «свободные бинарные диаграммы решений», в которых ограничение на порядок переменных заменяется на условие, что ни один путь из корневой вершины в терминальную не проверяет одну и ту же переменную более одного раза, «однократно ветвящиеся программы», которые обладают многими полезными свойствами OBDD, включая эффективный (хотя и вероятностный) метод проверки эквивалентности. В каждом из этих расширений метода, основанного на OBDD, делается попытка найти компромисс между компактностью представления и трудностью его построения или проверки его свойств.

Во многих задачах комбинаторной оптимизации символьные методы, использующие OBDD, не показали столь же высокую производительность, как более традиционные методы. Это связано с тем, что в этих задачах обычно требуется найти только одно решение, удовлетворяющее некоторому критерию оптимальности. Напротив, большинство основанных на OBDD подходов вычисляют все возможные решения и затем выбирают лучшее из них. К сожалению, многие задачи имеют слишком много решений, чтобы можно было их символически закодировать. Более традиционные методы поиска, такие, как метод ветвей и границ, часто оказываются более эффективными и способными решать большие задачи. Одна из возможностей улучшения состоит в использовании идеи «ленивого» или «отложенного» исполнения при обработке с использованием OBDD. Вместо того, чтобы немедленно создавать полное представление каждой функции в процессе выполнения последовательности операций, следует пытаться

строить только те OBDD, которые нужны для вывода окончательной информации.

5. ЗАМЕЧАНИЯ

Представление в виде упорядоченных бинарных диаграмм определяется путем наложения ограничений на бинарные диаграммы решений, введенные Ли [26] и Экерсом [6]. Эти ограничения и получаемая в итоге каноничность представления были впервые описаны Фортуном и др. [20]. Бинарные диаграммы решений много лет использовались в качестве абстрактного представления булевых функций. Под названием «программы ветвления» они были тщательно исследованы специалистами по теории сложности вычислений (см., например, Вегенер [37], Мейнел [30]). В своем изложении мы базируемся на работах Брянта [13-15], который ввел понятие OBDD в 1986 году. Дополнительная библиография по этой тематике может быть найдена в обзоре Мейнела и др. [31]. Более общее понятие логического фрагмента [3], которое позволяет представлять частичные функции, а также не только логические, но и другие функции от логических переменных, использовалось при исследовании схем Янова [2, 4].

Проблема нахождения оптимального упорядочения переменных для OBDD является NP-трудной [10, 36]. Эвристические методы, позволяющие находить хорошие способы упорядочения переменных для OBDD, предложены в работах Фуджита и др. [21], Малика и др. [28], Джонга и др. [22]. Свойство A доказано в работе [14], а свойство B — в работе [12], символьный симулятор COSMOS описан в [17]. Берман [8] и Макмиллан [29] вывели границы для нескольких классов сетей с «ограниченной шириной». Макмиллан [29] обобщил эту технику до *древовидного расположения*, в котором сеть организована в виде дерева логических блоков с коэффициентом ветвления b и первичным выходным значением, производимым корневым блоком.

Вопросы реализации OBDD на рабочих станциях с большой физической памятью рассматривались Брэйсом и др. в работе [11]. Другие исследователи [25, 33] успешно использовали для их реализации векторные операции и достигли хороших результатов при обработке OBDD на мультипроцессорах с разделенной памятью.

Усовершенствованиям базовой структуры OBDD, которые направлены на экономию памяти, посвящен целый ряд работ. Среди них — использование единственного многокорневого графа для представления всех требуе-

мых функций [13, 23, 32, 34], снабжение дуг метками для обозначения булевого вычитания [11, 23, 27, 32] и обобщение понятия областей с конечным числом состояний [35]. Ряд авторов рассматривал методы, основанные на тех же общих принципах, что и OBDD, но с менее строгими ограничениями структуры данных [7, 9, 16, 19, 23, 24, 37]. Возможности улучшения процессов обработки с использованием OBDD на основе идеи «ленивого» (или «отложенного») исполнения рассматривались в [5, 18].

СПИСОК ЛИТЕРАТУРЫ

1. **Евстигнеев В.А., Касьянов В.Н.** Теория графов: алгоритмы обработки бесконтурных графов — Новосибирск: Наука, 1998.
2. **Ершов А.П.** Введение в теоретическое программирование. — Новосибирск: Наука, 1977.
3. **Касьянов В.Н., Сабельфельд В.К.** Сборник заданий по практикуму на ЭВМ. — М.: Наука, 1986.
4. **Котов В.Е.** Введение в теорию схем программ. — Новосибирск: Наука, 1978.
5. **Abelson H., Sussman G. J., Sussman J.** Structure and interpretation of computer programs. — MIT Press, Cambridge, Mass, 1988. — P. 261–264.
6. **Akers S.** B. Binary decision diagrams // IEEE Trans. Comput. — 1978. — Vol. C-27, N 6. — P. 509–516.
7. **Ashar P., Devadas S., Ghosh A.** Boolean satisfiability and equivalence checking using general binary decision diagrams // The International Conference on Computer Design. — IEEE, New York , 1991. — P. 259–264.
8. **Berman C. L.** Ordered binary decision diagrams and circuit structure // The International Conference on Computer Design. — IEEE, New York, 1989. — P. 392–395.
9. **Blum M. W., Chandra A. K.** Equivalence of free Boolean graphs can be decided probabilistically in polynomial time // Inf. Process. Lett. — 1980. — Vol. 10, N 2. — P. 80–82.
10. **Bollig B., Wegener I.** Improving the variable ordering of OBDDs is NP-complete // IEEE Trans. Computers. — 1996. — Vol. 45. — P. 993–1002.
11. **Brace K. S., Bryant R. E., Rudell R. L.** Efficient implementation of a BDD package. In Proceedings of the 27th ACM / IEEE Design Automation Conference. — ACM, New York, 1990.— P. 40–45.
12. **Brayton R. K., Hachtel G. D., McMullen C. T., Sangiovanni-Vincentelli A. L.** Logic Minimization Algorithms for VLSI Synthesis. — Kluwer Academic Publishers, Boston, 1984.
13. **Bryant R. E.** Graph-based algorithms for Boolean function manipulation // IEEE Trans. Comput. — 1986. — Vol. C-35, N 6. — P. 677–691.

14. **Bryant R. E.** On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication // IEEE Trans. Comput. — 1991. — Vol. 40, N 2. — P. 205–213.
15. **Bryant R. E.** Simbolic Boolean manipulation with ordered binary-decision diagrams // ACM Computing Surveys. — 1992. — Vol. 24, N 3. — P. 293–318.
16. **Burch J. R.** Using BDDs to verify multipliers // Proc. of the 28th ACM / IEEE Design Automation Conference. — ACM, New York, 1991. — P. 408–412.
17. **Cho K., Bryant R. E.** Test pattern generation for sequential MOS circuits by symbolic fault simulation // Proc. of the 26th ACM / IEEE Design Automation Conference. — ACM, New York, 1989. — P. 418–423.
18. **Deguchi Y., Ishiura N., Yajima S.** Probabilistic CTSS: Analysis of timing error probability in asynchronous logic circuits // Proc. of the 28th ACM / IEEE Design Automation Conference. — ACM, New York, 1991. — P. 650–655.
19. **Finkbeiner B.** Language containment checking with nondeterministic BDDs // Lect. Notes Comput. Sci. — 2001. — Vol. 2031. — P. 24–38.
20. **Fortune S., Hopcroft J., Schmidt E. M.** The complexity of equivalence and containment for free single variable program schemes // Lect. Notes Comput. Sci. — 1978. — Vol. 62. — P. 227–240.
21. **Fujita M., Fujisawa H., Kawato N.** Evaluations and improvements of a Boolean comparison program based on binary decision diagrams // The Internat. Conf.on Computer-Aided Design. — IEEE, New York, 1988. — P. 2–5.
22. **Jeong S.-W., Plessier B., Hachtel G. D., Somenzi F.** Variable ordering and selection for FSM traversal // The Internat. Conf. on Computer-Aided Design. — IEEE, New York, 1991. — P. 476–479.
23. **Karplus K.** Using if-then-else DAGs for multi-level logic minimization // Advance Research in VLSI. — MIT Press, Cambridge, Mass., 1989. — P. 101–118.
24. **Katayama T., Ochi H., Tsuda T.** An algorithm for generating genetic BDDs // IEICE Trans. Fundamentals. — 2000. — Vol. E88-A, N. 12. — P. 2505–2512.
25. **Kimura S., Clark E. M.** A parallel algorithm for constructing binary decision diagrams // The Intern. Conf. on Computer Design. — IEEE, New York, 1990. — P. 220–223.
26. **Lee C. Y.** Representation of switching circuits by binary-decision programs // Bell System Tech. J. — 1959. — Vol. 38, N 4. — P. 985–999.
27. **Madre J. C., Billon J. P.** Proving circuit correctness using formal comparison between expected and extracted behaviour // Proc. of the 25th ACM / IEEE Design Automation Conference. — ACM, New York, 1988. — P. 205–210.
28. **Malik S., Wang A., Brayton K. K., Sangiovann-Vincentelli A.** Logic verification using binary decision diagrams in a logic synthesis environment // The Internat. Conf. on Computer-Aided Design. — IEEE, New York, 1966. — P. 6–9.
29. **McMillan K. L.** Symbolic model checking: An approach to the state explosion problem: PhD thesis, School of Computer Science, Carnegie-Mellon Univ, 1992.
30. **Meinel C.** Modified branching programs and their computational power // Lect. Notes Comput. Sci. — 1990. — Vol. 370.

31. **Meinel C., Stangier C.** Data structure for Boolean function. BDDs — foundations and aplications // Lect. Notes Comput. Sci. — 2001. — Vol. 370. — P.61–78.
32. **Minato S., Ishiura N., Yajima S.** Shared binary decision diagram with attributed edges for efficient Boolean function manipulation // Proc. of the 27th ACM / IEEE Design Automation Conf. — ACM, New York, 1990. — P. 52–57.
33. **Ochi H., Ishiura N., Yajima S.** Breadth-first manipulation of SBDD of function for vector processing // Proc. of the 28th ACM / IEEE Design Automation Conf. — ACM, New York, 1991. — P. 413-416.
34. **Reeves D. S., Irwin M. J.** Fast methods for switch-level verification of MOS circuits // IEEE Trans. CAD / IC CAD. — 1987. — Vol. 6, N 5. — P. 766–779.
35. **Srinivasan A., Kam T., Malik S., Brayton R. K.** Algorithms for discrete function manipulation // The Internat. Conf. on Computer-Aided Design. — IEEE, New York, 1990. — P. 92–95.
36. **Tani S., Hamaguchi K., Yajima S.** The complexity of the optimal variable ordering problems of shared binary decision diagrams // Lect. Notes Comput. Sci. — 1993. — Vol. 762. — P. 389–398.
37. **Wegener I.** On the complexity of branching programs and decision trees for clique functions // J. ACM. — 1988. — Vol. 35, N 2. — P. 461–471.

О. А. Лакийчук

АЛГОРИТМЫ ПОИСКА ДОМИНАТОРОВ В УПРАВЛЯЮЩЕМ ГРАФЕ*

1. ВВЕДЕНИЕ

Теория графов применяется в программировании с самого начала возникновения ЭВМ. Очень удобно выражать задачи обработки информации на теоретико–графовом языке. В заметке Р.Карпа (1960 г., на русском — 1962 г.) была введена в практику теоретико-графовая модель программы в виде управляющего графа. Эта модель стала к настоящему времени классической для решения задач трансляции и конструирования программ.

Современное состояние программирования нельзя представить себе без теоретико-графовых алгоритмов. В литературе можно найти описания огромного количества таких алгоритмов. Достаточно хотя бы рассмотреть книги “Теория графов : алгоритмы обработки деревьев” (1994) и “Сводимые графы и граф-модели в программировании” (1999). Авторы — В.А. Евстигнеев и В.Н. Касьянов. Для представления алгоритмов в этих книгах используется подход, базирующийся на языке высокого уровня (ВУ-язык) и модели машины с произвольным доступом к памяти (РАМ) [1, 2]. Мы также будем использовать этот подход, поскольку он позволяет формулировать алгоритмы работы с графами в естественной и наглядной форме, допускающей прямой анализ их корректности и сложности, а также довольно простой перенос на традиционные языки программирования.

В данной работе рассматриваются алгоритмы поиска обязательных предшественников (доминаторов) в управляющем графе. В разд. 2 описаны общие теоретические сведения, в разд. 3 представлены существующие алгоритмы поиска доминаторов. Разд. 4 посвящён новому алгоритму поиска доминаторов.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАБОТЫ

Дальнейшее изложение будет строиться на понятии управляющего графа [1–3].

Напомним некоторые сведения о нём. Транслируемая программа с точки зрения многих алгоритмов оптимизирующей трансляции рассматривается и обрабатывается как управляющий граф (также называют уграф, граф переходов, граф потока управления) — ориентированный граф, вершинами которого являются операторы программы, а дуги отражают возможность передачи управления между операторами при её исполнении.

Список задач анализа управляющих графов достаточно широк. Это задачи по определению структурных свойств графов (например, интервальная сводимость), достижению нужных структурных свойств графов (преобразование графа в сводимый, разрыв контуров и другие), задачи по декомпозиции графа на подграфы специального вида (гамаки, зоны). Здесь же рассматриваются задачи определения отношений доминирования и постдоминирования на уграфе.

Дальше под уграфом будем понимать ориентированный граф $G =(V,E,r)$ с выделенной начальной вершиной r *(вход)*, в которую не входит ни одна дуга. Каждая вершина G достижима из r. Вершина x *доминирует над* вершиной y *(x — обязательный предшественник* вершины y), если любой путь в G из s в y проходит через x. *Непосредственным доминатором* вершины w называется вершина v такая, что v доминирует над w и любой доминатор вершины w доминирует над v (обозначаем idom(w) = v). Отношение доминирования задаёт частичный порядок на V, а результирующее частично упорядоченное множество представляет собой ордерево, называемое *доминаторным деревом*.

3. ОПИСАНИЕ АЛГОРИТМОВ

3.1. Прямой алгоритм поиска доминаторов

Получается из определения доминируемости. Для каждой вершины $v \neq r$ выполняем следующие действия.

Основной шаг. Ищем множество всех вершин P, достижимых из r на путях, не содержащих вершину v. Множество $V\backslash\{v\}\backslash S$ состоит в точности из тех вершин уграфа, которые доминируются вершиной v.

Этот алгоритм имеет временную сложность $O(nm)$, где n — число вершин, m — число дуг.

3.2. Алгоритм Ленгауэра—Тарьяна

Более подробное описание алгоритма можно найти в [4].

Сначала введём некоторые понятия.

Пусть M(v) — M-номер вершины v и T — дерево поиска в глубину, тогда справедливо следующее свойство путей.

1. Пусть v,w — вершины. Если $M(v) \le M(w)$, то любой путь от v до w должен проходить через одного из общих предшественников вершин v и w в T.

Каждой вершине v сопоставляется семидоминатор *sdom(v)*, определяемый следующим образом:

$$Sdom(w) = \min\{M(v) \mid \text{существует такой путь } (v_0 = v, v_1, \ldots, v_k = w), \text{ что } M(v_i) > M(w) \text{ для всех I, } 1 <= I < k\}.$$

Дальше при формулировании свойств отождествляем имена вершин с их M-номерами. Используем обозначения $x \xrightarrow{*} y$, если x — предшественник y в T, а также $x \xrightarrow{+} y$, если $x \xrightarrow{*} y$ и $x \neq y$.

3.2.1. Свойства семидоминаторов

2. Для любой вершины $w \neq r$, $idom(w) \xrightarrow{+} w$, $sdom(w) \xrightarrow{+} w$, $idom(w) \xrightarrow{+} sdom(w)$.

3. Пусть v, w такие 2 вершины, что $v \xrightarrow{*} w$, тогда $v \xrightarrow{*} idom(w)$ или $idom(w) \xrightarrow{*} idom(v)$.

4. Пусть $w \neq r$ и каждая вершина v, для которой $sdom(w) \xrightarrow{+} v \xrightarrow{*} w$, удовлетворяет условию $sdom(w) \le sdom(v)$, тогда $idom(w) = sdom(w)$.

5. Пусть $w \neq r$ и v — вершина, которая имеет минимального семидоминатора sdom(v) среди вершин v таких, что $sdom(w) \xrightarrow{+} v \xrightarrow{*} w$, тогда $sdom(v) \le sdom(w)$ и $idom(v) = idom(w)$.

6. Пусть $w \neq r$ и v — вершина, которая имеет минимального семидоминатора sdom(v) среди вершин v таких, что $sdom(w) \xrightarrow{+} v \xrightarrow{*} w$, тогда

$$idom(w) = \begin{cases} sdom(w), & \text{если } sdom(w)=sdom(v) \\ idom(w), & \text{иначе} \end{cases}$$

7. Для любой вершины $w \neq r$ $sdom(w) = \min(\{v: (v, w) \in E, v < w\} \cup \{sdom(z) : z > w,$ существует $(v,w) \in E$ такая, что $z \xrightarrow{*} v\})$.

Доказательства этих свойств можно найти в [4].

С помощью рассмотренных свойств можно предложить быстрый алгоритм нахождения доминаторов. Он состоит из четырёх этапов.

1. Выполняется поиск в глубину. Вершинам приписываются их M-номера от 0 до n – 1, начиная с r. Проводится инициализация переменных.

2. Вычисляются семидоминаторы вершин уграфа по свойству 7. Обработка ведётся в порядке убывания M номеров.

3. По свойству 6 неявно вычисляются непосредственные доминаторы вершин.

4. Явно вычисляются idom(v) для каждой вершины v.

Каждой вершине сопоставим следующие атрибуты:

1) parent(v) — отец вершины v в дереве поиска в глубину T;
2) semi(v) — число, принимающее следующие значения:
   a) semi(v) = 0, пока v не получила своего M-номера,
   b) semi(v) = M(v) после получения вершиной v M-номера, но до вычисления idom(v),
   c) semi(v) = M(idom(v)) после вычисления idom(v);
3) bucket(v) — множество вершин, для которых v — доминатор;
4) dom(v) — вершина, определяемая следующим образом:
   a) после шага 3, если sdom(v) = idom(v), то dom(v) = idom(v), иначе dom(v) = x, где M(x) < M(v) и idom(x) = idom(v),
   b) после шага 4 dom(v) = idom(v).

Шаги 2 и 3 алгоритм выполняет одновременно. В процессе обработки вершин поддерживается временная структура данных, которая представляет собой лес, содержащийся в дереве поиска в глубину. Он образован множеством вершин V и дуг {(parent(w), w): w — уже обработана}. Для работы с этой структурой алгоритм использует 2 процедуры:

LINK(v,w) — добавляет в лес дугу (v,w);

EVAL(v) — возвращает v, если v — корень дерева в лесу. Иначе, пусть x — корень дерева в лесу, содержащего v. Eval возвращает любую вершину $u \neq r$ с минимумом semi(v) на пути $x \xrightarrow{*} v$.

При обработке вершины v алгоритм вычисляет семидоминатор вершины v, используя свойство 7; semi(v) = M(sdom(v)). После вычисления semi(v) алгоритм добавляет вершину v к множеству bucket(M^{-1}(semi(v))) и добавляет дугу в лес, используя вызов процедуры LINK(parent(v),v). Дальше выполняется шаг 3 для каждой вершины из множества bucket(parent(w)). Неявно вычисляется idom(v) по свойству 6. На шаге 4 алгоритм вычисляет явно idom(v), просматривая вершины в порядке возрастания их M-номеров.

Нетрудно доказать корректность алгоритма, используя свойства 6 и 7 при условии, что операции LINK и EVAL работают правильно.

3.2.2. Реализация операций LINK и EVAL

1. Простой способ — использует сжатие пути при выполнении операции EVAL. Для представления леса используются два атрибута вершин уграфа: ancestor и label. Сначала ancestor(v) = 0 и label(v) = v для любой вершины v. В общем, ancestor(v) = 0, если v — корень дерева в лесу, иначе ancestor(v) содержит отца v в лесу.

Значения атрибута label должны удовлетворять следующему свойству.

Пусть v — вершина, r — корень дерева в лесу, содержащего v, и $v = v_k, v_{k-1}, \ldots, v_0 = r$ — последовательность вершин такая, что ancestor(v_i) = v_{i-1} для всех i. Пусть x такая вершина, что semi(x) принимает минимальное значение среди вершин $x \in \{label(v_i): 1 \le i \le k\}$. Тогда semi(x) должно принимать минимальное значение среди вершин x, удовлетворяющих условию $r \xrightarrow{+} x \xrightarrow{*} v$.

При выполнении LINK(v,w) алгоритм присваивает ancestor(w) = v. Чтобы осуществить EVAL(v), алгоритм следует по ссылкам на отцов вершин для определения последовательности $v = v_k, v_{k-1}, \ldots, v_0 = r$; ancestor($v_i$) = v_{i-1} для всех i. Если v = r, то EVAL(v) возвращает v. Иначе алгоритм осуществляет сжатие пути, присваивая ancestor(v_i) = r для всех i от 2 до k, одновременно изменяя атрибуты label по правилу: если semi(label(v_{i-1})) < semi(label(v_i)), то label(v_i) = label(v_{i-1}). Затем EVAL(v) возвращает v.

2. **Сложный способ** — использует сжатие пути при выполнении EVAL, но реализует операцию LINK так, чтобы сжатие пути выполнялось только на сбалансированных деревьях.

3.2.3. Оценка временной сложности

Нетрудно показать, что временная сложность алгоритма Ленгауэра—Тарьяна составляет O(m + n) плюс время, необходимое для выполнения n – 1 операции LINK и m + n – 1 операции EVAL. Простая реализация этих

операций требует O(m * log n) времени. Такая же сложность и у всего алгоритма. В сложном случае время, требуемое для n – 1 операции LINK и m + n – 1 операции EVAL, составляет O(m * l(m, n)), где l — функция, обратная функции Аккермана.

4. НОВЫЙ АЛГОРИТМ

Проблема уменьшения временной сложности алгоритма поиска доминаторов — интересная задача. Этой проблемой занимались многие учёные. В 1985 г. Харел предложил почти линейный алгоритм [5], потом в 1997 г. Алструп усовершенствовал его [6]. Тарьян впервые использовал в своём алгоритме разбиение дерева поиска на микродеревья. Ниже описан новый алгоритм поиска доминаторов, использующий микродеревья и простые структуры данных. Он базируется на материалах работы [7].

Основные понятия такие же, как в алгоритме Ленгауэра—Тарьяна. При рассмотрении свойств вершины отождествляются со своими M-номерами.

4.1. Микродеревья

Рассмотрим следующую процедуру, помечающую вершины в дереве T. Параметр g задан, и все вершины изначально не помечены.

```
For x in D in reverse DFS order do
    S (x): = Σ (y child of x) S(y)
    If S (x)>g then
        Mark all children of x
    Endive
Done
Mark root (T)
```

Для каждой вершины v положим бпп(v) — ближайший помеченный предшественник. Функция бпп разбивает T на микродеревья следующим образом. Пусть v — помечена ; D(v) = {x|бпп(v) = v} — микродерево. root(D(v)) = v — корень микродерева D(v), micro(x) — микродерево, содержащее x (см. рис. 1).

Мы называем микродерево *нетривиальным*, если оно содержит хотя бы один лист дерева T. Все остальные микродеревья состоят ровно из одной

вершины и называются *тривиальными*. Вершина v, составляющая тривиальное микродерево, называется *специальной*, если каждый потомок v — корень нетривиального микродерева. На рис.1 c и e — такие вершины.

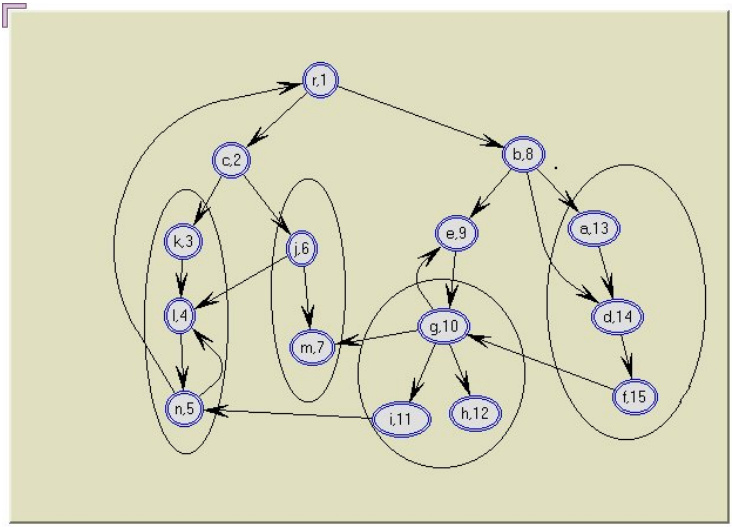


Рис. 1. Разбиение T на микродеревья величины не больше 3.
Корни — вершины k, j, g, a

4.2. Определение путей

Пусть P = (u = x_0, x_1, …, x_{k-1}, x_k = v) — путь в G. P — *внешний доминаторный путь* (XDOM путь), если P — Sdom путь и x_0, …, $x_{k-1} \notin$ micro(v).

Внешний доминатор вершины v:

$$xdom(v) = \min\{\{v\} \cup \{u| \text{ существует XDOM путь из u в v}\}\}.$$

Если v совпадает с тривиальным микродеревом, то xdom(v) = semi(v).
P — PXDOM путь, если $x_i \geq$ root(micro(v)), $1 \leq I \leq k–1$.

$$pxdom(v) = \min\{ u| \text{ есть PXDOM путь из u в v}\}.$$

Пример. Путь P = (c, j, l) — XDOM путь от c к l. Больше таких путей нет. idom(l) = 2. P = (r, b, e, g, I, n, l) — PXDOM путь. pxdom(l) = 1.

Справедливы следующие свойства.

2.1. Для каждой вершины, составляющей тривиальное микродерево, pxdom(v) = semi(v).

2.2. Если idom(v) $\notin$ micro(v), то idom(v) $\xrightarrow{*}$ pxdom(v).

4.3. Вычисление внутренних доминаторов

Для нетривиального микродерева введём понятие *расширенного графа* aug(D). Пусть G(D) — подграф уграфа G, построенный на вершинах микродерева D. aug(D) — граф G(D) плюс следующие дуги и вершины:

1) вершина t, называемая *корнем* aug(D) или root(aug(D));
2) дуга (t, v) для каждой вершины v $\in$ D такой, что есть дуга (u, v) $\in$ E для некоторой u $\notin$ D. Назовём эти дуги *синими* дугами.

Определяем внутреннего непосредственного предшественника вершины x — iidom(x) как idom(x) в aug(micro(x)).

Справедливы следующие свойства.

3.1. Если iidom(x) $\neq$ root(aug(micro(x))), то idom(x) = iidom(x).

3.2. Если iidom(x) = root(aug(micro(x))), то idom(x) $\notin$ micro(x).

4.4. Вычисление pxdom

Используем атрибут вершины label(v) в лесу (как в алгоритме Ленгауэра—Тарьяна), label(v) = v изначально.

Пусть EN(v) = {x | (x, v) $\in$ E, x $\notin$ micro(v)} — внешние соседи вершины v. Процедура работает в порядке убывания M-номеров.

1. Для v$\in$D:
   a) B = {label(x) | x $\in$ EN(v)};
   b) C = {label(eval(parent(root(micro(x))))) | x $\in$ EN(v), x не $\xrightarrow{*}$ v};
   c) Label(v) = min({v} $\cup$ B$\cup$C).
2. Пусть v$\in$D. Y(v) — множество вершин u из D таких, что есть путь из u в v, состоящий только из вершин G(D). Положим label(v) = min{label(y) | y $\in$ Y(v)}.
3. Если D — тривиальное микродерево, то выполняем link(v).

Справедливы следующие свойства.

4.1. После шага (1) label(x) = xdom(x) для всех x $\in$ D.

4.2. После шага (2) label(x) = pxdom(x) для всех x $\in$ D.

4.5 Вычисление доминаторов

Справедливы следующие свойства.

5.1. Для любой вершины v существует такая вершина $w \in micro(v)$, что

1) $w \xrightarrow{*} v$;
2) pxdom(v) = pxdom(w);
3) pxdom(w) = sdom(w);
4) iidom(w) = root(aug(micro(w))).

5.2. Пусть v и w — вершины в микродереве D такие, что

1) $w \xrightarrow{*} v$;
2) pxdom(v) = pxdom(w);
3) iidom(v) = iidom(w) = root(aug(T));

тогда idom(v) = idom(w).

Теперь представим сам алгоритм.

```
Algorithm IDOM
        For v ∈ D в порядке убывания M-номеров do
                Process(v)
        Done
    For u ∈ D, {u} — тривиальное микродерево, в порядке убывания M-
    номеров do
                Process(bucket(u))
                Link(u)
    Done
End Algorithm

Process(v)
    If iidom(v) ∉ micro(v) then
                Idom(v) = iidom(v)
    Else
                Add v to bucket(pxdom(v)) endif
End Process

Process(bucket(u))
    For v ∉ bucket(u) do
                If u = parent(root(micro(v))) then z = v
                Else z = eval(parent(root(micro(v))) endif
                If pxdom(z) = u then idom(v) = u
                Else idom(v) = idom(z) endif
  Done
End Process
```

Теорема. Алгоритм IDOM корректно вычисляет непосредственных предшественников вершин уграфа.

Доказательство. Свойство 3.1 показывает, что присваивание idom(v) = iidom(v) корректно, если iidom(v) $\in$ micro(v). Пусть теперь iidom(v) $\notin$ micro(v). Тогда idom(v) $\notin$ micro(v) по свойству 3.2.

Рассмотрим обработку вершины v из bucket(u). Пусть сначала pxdom(v) = sdom(v) = u. Пусть u1 — сын u на пути $u \xrightarrow{+} v$. Мы берем z — вершину на пути $u1 \xrightarrow{*} v$ с минимумом sdom и pxdom(z) = sdom(z). Это требование нужно, так как, если pxdom(z) = u, то по свойству 4 (алгоритм Л—Т) idom(v) = sdom(v) = u, и если pxdom(z) < u, то по свойству 5(Л—Т) idom(v) = idom(z).

Заметим, что для каждой w $\in$ micro(v) такой, что

$$w \xrightarrow{*} v,\ sdom(v) = pxdom(v) \le pxdom(w) \le sdom(w).$$

Таким образом, если u = parent(root(micro(v))), то u1 = root(micro(v)), z = v и требование выполняется.

С другой стороны, если

$$u \xrightarrow{+} parent(root(micro(v))),$$

то z = eval(parent(roor(micro(v)))) — вершина на пути

P = $u1 \xrightarrow{*}$ parent(root(micro(v))) с минимумом pxdom. Требование выполняется, так как (1) pxdom(u1) $\le$u = pxdom(v) и (2) pxdom(y) = sdom(y) для всех y $\in$ P (свойство 2.1).

Рассмотрим оставшийся случай, когда pxdom(v) $\ne$ sdom(v). По свойству 5.1 находим w. По предположению iidom(v) = root(aug(micro(x))), и idom(v) = idom(w) по свойству 5.2. Так как pxdom(v) = pxdom(w) и micro(v) = micro(w), то v и w лежат в одном bucket. Поэтому, исходя из предыдущих рассуждений, алгоритм правильно вычисляет значение idom(w), что и требовалось доказать**.**

4.6. Анализ временной сложности

Простой анализ показывает, что при g = $O(\log^{1/3} n)$ вычисление iidom требует порядка O(n + m) времени. Вычисление pxdom требует также O(m + n) времени плюс время на выполнение операций link и eval. Нетрудно показать, что тогда общее время алгоритма $O(m * l(m, n/\log^{1/3} n) + n)$. Известно, что в этом случае $l(m, n/\log^{1/3} n) = O(1)$, а значит алгоритм работает за линейное время O(m + n).

5. ЗАКЛЮЧЕНИЕ

В статье рассмотрены некоторые существующие алгоритмы поиска доминаторов в управляющем графе и представлен новый линейный алгоритм.

СПИСОК ЛИТЕРАТУРЫ

1. **Евстигнеев В.А., Касьянов В.Н.** Теория графов: алгоритмы обработки деревьев. — Новосибирск: Наука, 1994.
2. **Евстигнеев В.А., Касьянов В.Н**. Сводимые графы и граф — модели в программировании. — Новосибирск: ИДМИ, 1999.
3. **Ахо А., Ульман Дж.** Теория синтаксического анализа, перевода и компиляции. Т. 1, 2. — М.: Мир, 1978.
4. **Lengauer T., Tarjan R.E.** A fast algorithm for finding dominators in a flow graph // ACM Trans. Program. Lang. Syst. — 1979. —Vol. 1, N 1. — P.121–141.
5. **Harel D.** A linear time algorithm for finding dominators in flow graphs and related problems // Proc. of the 17th ACM Sympos. on Theory of Computing. –ACM Press, N.-Y., 1985. — P. 185–194
6. **Alstrup S., Harel D.** Dominators in linear time. — 1997. `ftp://ftp.diku.dk/pub/diku/users/stephen/dom.ps`
7. **Buchsbaum A. L., Kaplan H., Rogers A., Westbrook J. R.** A new, simpler linear-time dominators algorithm // ACM Trans. Progr. Lang. and Systems. — 1998. — Vol. 20, N 6. — P. 1265–1296.

Ю. В. Малинина

ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ АВТОМАТИЧЕСКОГО ФОРМИРОВАНИЯ ТЕМАТИЧЕСКОЙ КОЛЛЕКЦИИ ПО ПРЕОБРАЗОВАНИЯМ ПРОГРАММ ДЛЯ КОЛЛЕКТИВНОГО ИСПОЛЬЗОВАНИЯ*

ВВЕДЕНИЕ

В настоящее время происходят революционные процессы в изменении структуры мировой системы научных коммуникаций. В связи с этим можно отметить реферативный журнал, созданный институтом информации Гарфильда “Science Citation Index”, ориентированный на поиск новых научных публикаций в мировой системе периодических и продолжающихся изданий по системе научных ссылок. Это издание использует естественную, исторически сложившуюся систему классификации научных работ по ссылкам автора на работы его предшественников. В нем отсутствует разбиение статей на тематические рубрики, авторы указателя предлагают пользователям подготовленные ими тематические кластеры связанных между собой публикаций по системе цитирования общих предшественников. Как размеры, так и наименования кластеров постоянно корректируются ими в соответствии с тенденциями в науке.

Следующим шагом в развитии коммуникации в мировом научном сообществе стало широкое распространение электронной почты, позволившей многим ученым реализовать каналы неформальной коммуникации, которая ранее могла происходить только на конференциях, симпозиумах и семинарах. Это привело к образованию большого числа “невидимых колледжей”, неформальных объединений ученых, работающих в одной тематической области науки. Этот процесс неформальной коммуникации, происходящий в настоящее время, еще достаточно подробно не изучен и ждет своих исследователей.

Наиболее значимым событием в развитии системы научной коммуникации явилось появление в Internet (мировой информационной сети) информационных страниц различных научных школ (университетов, научных

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

институтов, коллективов и т. д.) и даже отдельных ученых. Одним из важнейших событий стало представление в них библиографий научных документов (публикаций в мировой научной печати, сообщений, тезисов и т. д.), созданных в рамках их научных исследований. Многие ученые поняли важность такой информационной работы, позволяющей сохранить для следующих поколений исследователей результаты их научной работы.

Наиболее ценными информационными объектами среди них можно считать списки полных библиографий научных документов, относящихся к одной тематической области, которые создавались энтузиастами-учеными и целыми международными коллективами в рамках исследовательских проектов.

Задача автоматического создания и дальнейшего пополнения подобной тематической коллекции в области преобразований программ рассматривается в данной статье.

СБОР ИНФОРМАЦИИ

Для получения информации в Интернете в основном используются поисковые системы общего назначения, которые выполняют индексирование всех существующих страниц. Построение индексов Интернет-ресурсов для поисковых систем традиционно основано на использовании сетевых роботов-программ, которые, начиная с некоторой Интернет страницы, рекурсивно обходят ресурсы Интернет, извлекая ссылки на новые ресурсы из получаемых документов. В каждом случае используется некоторый определенный метод извлечения информации.

Особенности задачи поиска в Интернете

Методы поиска, используемые в классических поисковых системах, разрабатывались и тестировались на относительно небольших, однородных коллекциях документов, например, таких как библиотечные каталоги или коллекции газетных статей. Интернет как набор данных имеет ряд важных особенностей.

1. Данные в Интернете организованы крайне стихийно и не систематично. Несмотря на то что принято считать, что Интернет — это распределенный гипертекст, это не совсем так. Гипертекст обычно подразумевает наличие концептуальной модели, которая накладывает ограничения согласованности на данные и гиперсвязи. В Интернете это обычно не так

даже для тех его частей, которые находятся под единым административным контролем. Около 30% информации в Интернете составляют точные или приблизительные копии других документов.

2. Текущий объем доступной информации в Интернете оценивается в десятки терабайт и быстро возрастает, поэтому даже самый мощный сетевой робот не может отслеживать содержание всех ресурсов Интернета. Отметим, что эти оценки касаются только той «поверхностной» части Интернета, которая не скрыта за поисковыми формами и доступ к которой не требует предварительной регистрации или авторизации. Другую, «скрытую», часть Интернета (hidden web) поисковые системы обычно не рассматривают, а ведь к ней относится множество крупных баз данных, опубликованных в Интернете. Поэтому неудивительно, что оценка объема «скрытого» Интернета в 500 раз больше, чем объем «поверхностного» Интернета.

3. По некоторым оценкам ежемесячно публикуется около 30 миллионов новых документов, причем ежемесячно изменяется до 40% ранее опубликованной информации. Среднее время жизни Интернет-страницы около 24 дней.

4. Большая доля документов виртуальна в том смысле, что формируется в ответ на некоторый запрос пользователя и не хранится в явном виде. Как правило, это результаты поиска в различных базах данных.

5. В Интернете используются более 100 естественных языков. Сами документы представляются в различных форматах, таких как html, xml, Word, PostScript, PDF и т.п.

6. Отсутствие редакторского контроля над публикуемой информацией в Интернете обуславливает проблему с ее качеством: информация может быть некорректной (например, уже устаревшей), ложной, плохо сформулированной, содержать массу ошибок (опечаток, грамматических ошибок, ошибок оцифровки и т.п.). Так, по некоторым оценкам, одна опечатка встречается в среднем в каждых двухстах часто употребительных словах или в трех иностранных фамилиях.

Особенности структуры Интернета

Интуитивно кажется естественным предположение о том, что ссылки с некоторой Интернет-страницы в основном ведут на страницы близкой тематики. Эмпирическое подтверждение согласованности между тематической и пространственной (в смысле расстояний в графе Интернет-страниц) локальностью было дано в работе [4].

«Тематическое Сообщество» можно определить как совокупность страниц, каждая из которых имеет больше ссылок (в любом направлении) внутри этой совокупности, чем снаружи. Данное определение можно обобщать, для того чтобы выявить объединения различного размера с различным уровнем связности. Более строгое определение и алгоритм выявления таких сообществ можно найти в [6].

Подходы к организации поиска

Существуют два основных подхода. Первый из них заключается в использовании индексов существующих универсальных поисковых систем. Этот подход достаточно широко применяется на практике и имеет свои как положительные, так и отрицательные стороны.

К его положительным сторонам относятся:

- повторное использование ранее полученных данных. Сканирование Интернета — это дорогостоящий процесс, который приводит не только к большим затратам для проводящей его организации, но и затрагивает другие интересы владельцев индексируемых сайтов, пользователей. Представляется весьма неразумным многократно сканировать Интернет ради предоставления доступа к одной и той же информации из большого числа конкурирующих поисковых систем;
- новизна используемой информации. Универсальные поисковые системы стремятся (в идеале) индексировать все новые документы, не ограничивая себя фиксированной и, возможно, уже устаревшей тематикой. Индекс таких систем предоставляет представительную выборку относительно недавно опубликованных документов, что можно использовать при анализе тенденций в той или иной области;
- низкая стоимость получения информации. Фактически на данный момент получение информации в виде ответов на автоматически

генерируемые запросы от коммерческих поисковых систем бесплатно.

Отрицательные стороны представлены как:

- старение индекса;
- закрытость методики получения используемой информации. Алгоритмы сканирования Интернета и поиска в индексе являются коммерческой тайной, и, следовательно, индекс коммерческих универсальных систем представляется используемым его системам следующего уровня в виде черного ящика;
- невозможность делать какие-то объективные выводы о характере распределения информации в Интернете на основе косвенного (через систему поиска) анализа индекса коммерческой системы;
- ненадежность доступа к информации. Коммерческие поисковые системы очевидно не заинтересованы в том, чтобы их индекс анализировался какими бы то ни было автоматическими системами. В любой момент эксперименты в этой области могут быть запрещены в связи с нарушением тех или иных прав коммерческих поисковых систем.

Второй подход к реализации поиска связан с самостоятельным обходом Интернета и основан на использовании ссылок на новые документы из ранее загруженных документов.

Традиционно основой информационных систем является сетевой робот — это программа, которая, начиная с некоторой Интернет-страницы, рекурсивно обходит ресурсы Интернета, извлекая ссылки на новые ресурсы из получаемых документов. Для этого процесса используется метафора «ползанья» паука применительно к созданию гиперссылок в Интернете. Этот процесс повторяется с каждым новым набором страниц и продолжается до тех пор, пока не перестанут появляться новые страницы либо пока не будет собрано предопределенное количество страниц.

Фактически большинство сетевых роботов не может посещать все доступные в Интернете ресурсы из-за ограниченности доступных роботу аппаратных и сетевых ресурсов, и то, какие именно ресурсы будут посещены, определяется применяемой стратегией посещения. Естественно, что робот должен стараться использовать такую стратегию, которая максимизирует общую «полезность» всех посещенных ресурсов. Все обнаруженные, но не просмотренные страницы помещаются роботом в приоритетную очередь, упорядоченную по качеству. «Полезность» ресурса определяется той целью, для достижения которой используется робот.

Робот, который собирает информацию о ресурсах для поисковой системы общего пользования, заинтересован в обнаружении максимального количества разнообразных ресурсов. Подобные роботы зачастую используют в качестве оценки «полезности» ресурса глубину URL, т. е. количество промежуточных каталогов, упоминающихся в URL между именем Интернет-узла и именем самого ресурса. Чем больше глубина, тем ниже важность соответствующего ресурса. Такой подход позволяет быстро посетить стартовые и близкие к ним страницы на большом числе Интернет-узлов.

Остановимся на положительных сторонах второго подхода, к которым относятся:

- объективность. В данном случае информация извлекается непосредственно из сети, что обеспечивает ее объективность;
- управляемость процесса получения информации. В отличие от косвенного доступа к индексу коммерческой системы через формирование специальных запросов, доступ к информации в данном подходе более прозрачен. Имеется прямая и легко обнаруживаемая связь между алгоритмом сканирования Интернет и тематической направленностью загружаемых документов. Это позволяет подбирать параметры алгоритма, минимизирующие среднее отклонение тематики загружаемых документов от заданного тематического направления.

Отметим и отрицательную сторону второго подхода:

- высокая стоимость. В данном случае сетевой робот реально сканирует Интернет, что приводит к большим затратам даже при использовании ограниченного (заданной тематикой) поиска.

Однако давно замечено, что эффективность одного и того же метода поиска часто варьируется при применении его в различных коллекциях. Сопоставление таких наблюдений и характеристик коллекций зачастую позволяет выявить слабые места и, как следствие, способствует повышению эффективности методов поиска.

Естественным развитием этой идеи является определение характеристик коллекций документов с целью применения этой информации для повышения эффективности поиска. Это свойство позволяет рассматривать задачу автоматического формирования тематической коллекции как перспективное направление.

За последние несколько лет этой проблеме было посвящено много внимания. В данной работе рассматриваются подходы к автоматическому построению тематической коллекции в области преобразований программ.

СОСТАВЛЕНИЕ ТЕМАТИЧЕСКОЙ КОЛЛЕКЦИИ.

Рассмотрим задачу формирования тематической коллекции. Традиционно считается, что пользователь использует систему информационного поиска для обнаружения документов, содержащих информацию на связанную с его запросом тему. Однако такое предположение верно не всегда. Поскольку представление пользователя о том, что такое релевантный документ, напрямую зависит от цели, для достижения которой он проводит поиск, то естественной кажется идея оптимизировать метод поиска под конкретную цель пользователя.

В нашем случае преследуются следующие цели.

Выявление сообществ

Тематическая коллекция документов, ориентированная на определенную предметную область и формируемая на основе общедоступного Интернета, должна использовать механизмы поиска, ориентированные на определенную предметную область, и иметь возможность определять подмножества Интернета в рамках своей предметной области. Главной задачей применяемой стратегии посещения документов является выбор такого порядка обхода известных роботу ресурсов, при котором за минимальное время будет обнаружено максимальное число документов, релевантных тематике.

Составление обзора

При составлении обзора пользователю недостаточно просто найти документы с информацией по соответствующей теме. Для адекватного представления необходимо, чтобы тематическая коллекция содержала информацию с разными точками зрения. Если не учитывать специфику этой задачи поиска, то вероятно, что обнаруженные документы будут отражать только одну, доминирующую в ресурсах Интернет, точку зрения.

Один из возможных подходов к решению этой задачи состоит в построении по исходному запросу и множеству возвращаемых документов так называемого «обратного» запроса, ориентированного на получение списка документов, отражающих другие точки зрения. Для этого в исходный запрос могут быть добавлены ссылки на экспертов, которые отражают еще не найденные точки зрения, или наоборот исключены/запрещены ссылки на уже представленных экспертов. Информация об экспертах, выражающих еще не обнаруженные точки зрения по этому вопросу, может быть почерп-

нута, например, из специализированного тезауруса. Отметим, что здесь термин «эксперт» вовсе необязательно означает «человек».

Поиск по категории

Еще одним типичным примером изменения цели поиска является сужение области поиска на документы определенной категории, такой как, например, множество домашних страниц специалистов в области преобразований программ или множество анонсов научных конференций. Прямолинейным подходом к решению этой проблемы является поиск в соответствующем разделе составленного вручную каталога типа Yahoo!, но соответствующий искомой категории раздел не всегда существует, да и с большой вероятностью он содержит ссылки на далеко не все ресурсы этой категории.

Одним из возможных подходов к реализации поиска по категории является выявление различных атрибутов, характеризующих страницы данной категории, и использование этой информации для расширения запроса. Эти атрибуты далеко не всегда очевидны: так, например, при поиске по категории англоязычных домашних страниц наивное расширение запроса путем добавления «home page» повышает точность на 20%, а менее очевидные, но автоматически построенные расширения «my» и «welcome» повышают точность на 65%.

Еще один вариант — фильтрация результатов поиска. Наиболее очевидной его реализацией является применение методов бинарной классификации для определения, какие из найденных документов относятся к заданной категории.

Поиск научных публикаций

В Интернете доступно огромное количество научных работ, многие из которых еще не доступны в печатном виде, и это мотивирует интерес к задаче поиска по научной литературе.

Конечно, в некотором роде эта цель является частным случаем цели поиска по категории (см. выше). Однако этот случай выделен по нескольким причинам. Во-первых, научные статьи зачастую хранятся в форматах отличных от html. Во-вторых, поскольку научная литература гораздо больше похожа на публикации в том смысле, как этот термин понимается в библиотеках, то здесь возможно применение подходов из библиотечной области. Одним из очень полезных механизмов является использование «индекса цитирования», т.е. количества библиографических ссылок на данную работу из других статей.

МЕТРИКИ

Качество работы агента и скорость нахождения важных ссылок определяются качеством используемых фильтров. Несмотря на подтверждение того, что документы в Интернете имеют тематическую локальность для многих общих тематических областей, нет ясного понимания того, как это использовать для конкретных условий. Кроме того, в чистом виде этот подход не использует большую часть доступной информации как, например, точное содержание не просмотренных Интернет-страниц или структуру URL-кандидата. Такие данные могут обеспечить ценную информацию, для того чтобы направлять обход более эффективно. Можно ожидать, что в общем случае один из этих показателей может оказаться более важным и что упорядочение на основании дополнительных показателей может существенно корректировать направление обхода. Поэтому кажется целесообразным использовать некоторую совокупность показателей.

Примем следующие соглашения: набор Интернет-страниц рассматривается в виде направленного графа G, где каждая Интернет-страница в наборе моделируется узлом графа; если страница p_1 содержит гиперссылку на страницу p_2, то в графе есть направленное ребро (p_1, p_2); если p_1 не имеет гиперссылки на p_2, то направленного ребра (p_1, p_2) не существует.

Метрика цитируемости PR(p)

Простейшая идея глобального (т.е. статического) учета ссылочной популярности состоит в подсчете числа ссылок, указывающих на страницу. Это примерно то, что в традиционной библиографии называют индексом цитирования.

Первое допущение методов на основе связей порождает простой критерий: чем больше гиперссылок указывает на страницу, тем лучше эта страница. Основной недостаток такого подхода состоит в том, что он не делает различий между качеством страницы, на которую указывает несколько страниц низкого качества, и качеством страницы, на которую указывает то же число страниц высокого качества. Очевидно, что рейтинг страницы можно увеличить, просто создав множество других страниц, ссылающихся на данную страницу. Поэтому обычно используется модификация этой метрики, предложенная Брин и Пейдж как алгоритм PageRank [3, 4]. Он вычисляет коэффициент PageRank каждой страницы, присваивая каждой ссылке на страницу весовой коэффициент, пропорциональный качеству страницы, содержащей гиперссылку. Чтобы определить качество ссылаю-

щейся страницы, используются ее коэффициенты PageRank рекурсивно, причем первоначальные значения PageRank задаются произвольно. Точнее, $PR(p_1)$ — коэффициент PageRank страницы p_1 можно определить как

$$PR(p_1) = \frac{a}{n} + (1-a)\sum \frac{PR(p_{2)}}{outegree(p_2)},$$

где a — константа, значение которой находится в пределах от 0,1 до 0,2; n — число вершин, т.е. число Интернет-страниц в наборе; outdegree(p_2) — число ребер из p_2, т.е. число гиперссылок.

Эта формула показывает, что коэффициент PageRank страницы A зависит от PageRank страницы B, указывающей на A. Поскольку определение PageRank порождает такое линейное уравнение для каждой страницы, чтобы вычислить PageRank для всех страниц, необходимо решить огромное количество линейных уравнений. PageRank позволяет эффективно отличить высококачественные страницы Интернета от низкокачественных.

Метрика концентрации и авторитетности HR(p), AR(p)

Если предположить, что информация по теме может распределиться примерно поровну между страницами с хорошим информационным наполнением по теме, называемыми «авторитетами» (authority), и страницами, напоминающими каталоги, с множеством ссылок на другие страницы, посвященные данной теме, называемыми «концентраторами» (hub), то алгоритм Кляйнберга — поиск документов по заданной теме на базе гиперссылок (Hyperlink-Induced Topic Search — HITS) — пытается выявить хорошие концентраторы и авторитеты [7, 8]. Алгоритм итеративно вычисляет показатель концентрации и авторитетности для каждого узла графа соседей, а затем упорядочивает узлы в соответствии с этими показателями. Узлы, имеющие высокие показатели авторитетности, должны быть хорошими авторитетами, а узлы с высокими показателями концентрации должны быть хорошими концентраторами. Алгоритм исходит из того, что документ, ссылающийся на большое число других документов, — хороший концентратор, а документ, на который указывает множество других документов, — хороший авторитет. Рекурсивно документ, который указывает на большое число хороших авторитетов, — еще лучший концентратор, а документ, на который ссылается множество хороших концентраторов, — еще лучший авторитет.

Заметим, что алгоритм не утверждает, что будут найдены все высококачественные страницы, удовлетворяющие условиям запроса, поскольку некоторые из таких страниц могут не принадлежать графу соседей или входить в его состав, но не иметь ссылок со многих страниц.

Дополнительно следует учитывать, что в связи с алгоритмом HITS возникает две проблемы.

1. Поскольку рассматривается относительно небольшая часть графа Интернета, добавление ребер к нескольким узлам может серьезно изменить показатели концентраторов и авторитетов [10]. В силу этого манипулировать этими показателями достаточно просто, поэтому манипуляция ранжированием механизма поиска — серьезная проблема для Интернета.

2. Если большая часть страниц в графе соседей относится к теме, которая отличается от темы запроса, авторитеты и концентраторы, получившие высокий рейтинг, могут относиться к другой теме. Эта проблема называется «смещением темы». Добавление весов к ребрам с учетом текста документов или их тезисов значительно смягчает негативный эффект этой проблемы.

Метрика местоположения LR(p)

Метрика местоположения отражает зависимость значения страницы от ее местонахождения. Если URL заканчивается на ".com", то страница может считаться более полезной, чем URLs с другими окончаниями, или URLs содержит вхождение "home", то она может быть более интересна, чем другие URLs.

Метрика тематического соответствия IR(p, F)

Рассматривается многомерное векторное пространство, где каждому термину соответствует свое измерение. Тематический фильтр представляет собой *вектор значимостей терминов фильтра* $(F_1,...,F_n)$. Каждый документ p представлен *вектором значимостей терминов* $(p_1,...,p_n)$ в этом векторном пространстве. Формула для вычисления значимости термина (p) в документе с учетом косинусного фактора нормализации представляется формулой

$$p_i = \frac{tf_i \times idf_i}{\sqrt{\sum_i p_i^2}},$$

где tf_i (term frequency) — частота, с которой встречается данный индексный термин; idf_i (inverted document frequency) — величина, обратная частоте, с которой данный термин встречается во всей совокупности документов.

Вхождения терминов, встречающихся в документе, положительны: $p_i > 0$, а вхождения терминов, не встречающихся в документе, равны нулю: $p_i = 0$. Величина p_i представляет собой функцию, которая растет в зависимости от частоты употребления термина в документе и убывает в зависимости от числа документов в наборе, содержащем этот термин. Идея состоит в том, что чем больше документов, в которых присутствует данный термин, тем в меньшей степени термин характеризует данный документ, и чем чаще термин встречается в документе, тем в большей степени он характеризует документ.

Адекватность документа фильтру (релевантность) вычисляется как скалярное произведение их векторов терминов. В качестве результата документы упорядочиваются в порядке убывания их показателей

$$IR(p, F) = \sum p_i - F_i.$$

Многие авторы Интернет-страниц заинтересованы в том, чтобы их страницы в ответах на определенные запросы имели высокий рейтинг. Таким образом, чтобы увеличить рейтинг, авторы будут различным образом «подкручивать» свои страницы. Эти попытки манипулирования алгоритмом ранжирования иногда доходят вплоть до того, что добавляются фрагменты текста, набранные невидимым шрифтом. Например, если для ранжирования используется модель векторного пространства, добавление на страницу 1000 слов «машина» увеличит рейтинг данной страницы при поиске по запросу «машина».

Любой алгоритм, базирующийся исключительно на содержимом страницы, восприимчив к такого рода манипуляциям. Действенность анализа гиперссылок проявляется в том, что он использует для определения рейтинга текущей страницы информационное наполнение других страниц. Если предположить, что эти страницы были созданы авторами независимо от автора исходной страницы, то объективность упорядочивания при таком подходе растет.

СОСТАВЛЕНИЕ КОЛЛЕКЦИИ

Задача агента, предлагаемого в данной работе, состоит в пополнении коллекции новыми релевантными ее тематике документами. Используя описанные выше метрики, составление коллекции будет включать в себя следующие основные этапы.

1. Генерация фильтра коллекции.

Фильтр коллекции строится на основе анализа содержимого ядра коллекции. В момент анализа для всех слов из словаря коллекции (слова, встречающиеся в ее ядре) вычислялись веса. В качестве фильтра коллекции выбирается заданное количество слов из словаря коллекции с наибольшими весами.

2. Инициализация дерева URL.

На этапе инициализации формируется дерево с двумя уровнями. На первом уровне — корень дерева, на втором — узлы, содержащие стартовые URL, заданные администратором коллекции. Каждому узлу v приписывается оценка $P(v)$ вероятности того, что ссылка из соответствующего документа указывает на документ релевантный тематике коллекции

$$\mathrm{P(v)} = \frac{1}{4}(\mathrm{HR(v)} + \mathrm{AR(v)} + \mathrm{PR(v)} + \mathrm{LR(v)} + \mathrm{IR(v, F)}).$$

Для стартовых URLs на этапе инициализации эти оценки принимаются равными 1.

3. Выбор URL (этот пункт и последующие повторяются в течение всего времени работы агента).

В дереве URL выбирается узел v, для которого $P(v)$ максимально (при этом не выбираются URLs, указывающие на недавно посещенные сайты). Если документ с соответствующим URL еще не загружен, то для последующей загрузки выбирается данный URL. В противном случае, случайным образом выбирается еще не рассмотренная ссылка из этого документа на новый документ в формате html. Если нерассмотренных ссылок нет, то выбирается другой узел.

4. Загрузка и фильтрация документа.

С помощью программы wget загружается документ с заданным URL. Выполняется разбор текста документа, в процессе которого выделяются ссылки на другие html-документы, и вычисляется *tf*-вектор загруженного документа.

5. Модификация дерева URL.

Если документ не прошел фильтрацию, то он не рекомендуется для включения в коллекцию. Если для URL данного документа в дереве URL уже имеется узел, то он помечается как нерелевантный и оценка вероятности релевантности исходящих из него ссылок принимается равной нулю. В противном случае, оценка $P(v)$ уменьшается.

Если документ прошел фильтрацию, то он рекомендуется для включения в коллекцию. Если для URL данного документа в дереве URL уже имеется узел, то он помечается как релевантный и оценка вероятности релевантности исходящих из него ссылок принимается равной 1. В противном случае, оценка $P(v)$ увеличивается (если оно было меньше 1). Кроме того, формируется новый узел w, содержащий URL загруженного документа. Величина $P(v)$ принимается равной 1. В дереве URL узел w является сыном узла v.

Новое значение $P(v)$ равно

$$P(v) = \frac{1 + links_{+}(v)}{1 + links_{+}(v) + links\_(v)},$$

где $links_{+}(v)$ равно числу проверенных релевантных ссылок из документа в узле v, а $links\_(v)$ — числу проверенных нерелевантных ссылок из того же документа.

ЗАКЛЮЧЕНИЕ

Хотя на данный момент эксперименты не позволяют делать статистически значимых выводов, но в дальнейших исследованиях предполагается получить улучшенную стратегию формирования тематической коллекции.

Механизмы поиска позволяют быстро и просто получить доступ к огромным объемам информации. Их вклад в развитие Интернета и общества в целом трудно переоценить. Однако «универсальная» модель поиска в Интернете зачастую ограничивает разнообразие и полезность получаемых результатов. Предотвратить это может более активное использование контекста и тематической направленности при поиске в Интернете.

СПИСОК ЛИТЕРАТУРЫ

1. **Bharat K., Henzinger M.R.** Improved algorithms for topic distillation in a hyperlinked environment // Research and Development in Information Retrieval: Proc. / SIGIR'98: 21st Annual Internat. ACM SIGIR Conf., Melbourne, Australia, August, 1998. — ACM, 1998. — P. 104–111
2. **Chakrabarti S., Berg M. van den, Dom D.** Focused crawling: A new approach to topic-specific Interrnet resource discovery // Searching and Querying: Proc. / 8th World Wide Web Conf., Toronto, May 1999 (http://www8.org/w8-papers/5a-search-query/crawling/index.html).
3. **Cho J., Garcya-Molina H., Page L.** Efficient crawling through URL ordering // Search and Indexing Techniques II: Proc. / 7th World-Wide Web Conf., Brishbone, Australia, April, 1998 (http://www7.scu.edu.au/programme/fullpapers/1919/com1919.htm).
4. **Davison B. D.** Topical locality in the Web // Research and Development in Information Retrieval: Proc. / SIGIR'00: 23rd Annual Internat. ACM SIGIR Conf., Athens, Greece, July 2000. — ACM, 2000. — P. 272–279 (http://citeseer.nj.nec.com/271585.html).
5. **Diligenti M., Coetzee F, Lawrence S., and other.** Focused crawling using context graphs //Very Large Data Bases: Proc. / VLDB 2000: 26th Internat. Conf., Cairo, Egypt, September 2000 — Morgan Kaufmann, 2000. — P 527–534 (http://www.informatik.uni-trier.de/~ley/db/conf/vldb/DiligentiCLGG00.html).
6. **Flake G., Lawrence S., Giles C. L.** Efficient Identification of Web Communities // Knowledge Discovery and Data Mining: Proc. /SIGKDD'00: 6th ACM Int'l Conf., Boston, MA, August 2000. — ACM, 2000. — P. 150–160.
7. **Kleinberg J., Gibson D., Raghavan P.** Inferring Web communities from link topology // Hypertext and Hypermedia: Proc. / 9th ACM Conf., Pittsburgh, Pennsylvania, USA, June, 1998. — ACM, 1998. — P 225–234 (http://citeseer.nj.nec.com/36254.html).
8. **Kleinberg J.** Authoritative sources in a hyperlinked environment // Discrete Algorithms : Proc. / ACM-SIAM Symp., San Francisco, California, USA, January, 1998. — ACM 1998. — P. 668–677.
9. **Лоуренс С.** Контекст при поиске в Интернет // Открытые системы. — 2000. — N 12. — (http://www.osp.ru/os/2000/12/062.htm).
10. **Хензингер М.** Анализ гиперссылок в Web // Открытые системы. — 2001. — N 10. — (http://www.osp.ru/os/2001/10/050.htm).
11. **Романова Е.В., Романов М.В., Некрестьянов И.С**. Использование интеллектуальных сетевых роботов для построения тематических коллекций // Тр. 1-й Всерос. науч. конф. «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». — Санкт-Петербург, 1999 (http://www.dl99.nw.ru/PDF/16.pdf).

Д. Ю. Мехонцев, И. В. Лобив, Ф. А. Мурзин

РЕШЕНИЕ ЗАДАЧИ НАХОЖДЕНИЯ ОПТИМАЛЬНОГО ПОЛОЖЕНИЯ ТЕЛА В ПРОСТРАНСТВЕ ПО ДАННЫМ, ПОСТУПАЮЩИМ С ОДНОМЕРНЫХ КАМЕР, ДЛЯ 3D ОПТИЧЕСКОЙ СИСТЕМЫ АНАЛИЗА ДВИЖЕНИЯ ОБЬЕКТОВ*

ПОСТАНОВКА ЗАДАЧИ

В пространстве имеется твердое тело с известной геометрией, на котором в определенных местах закреплены светоизлучающие диоды, в дальнейшем называемые маркерами, которым присвоены уникальные номера. Также имеется набор одномерных детекторов, расположенных вокруг твердого тела, которые достаточно часто (200–300 раз в секунду) фиксируют положение маркеров. Каждый маркер обладает уникальными характеристиками (цвет, частота мерцания, и т.п.), по которым детектор может отличить его от других маркеров и восстановить его номер. В силу одномерности детектора о каждом маркере, попадающем в его поле видимости, можно получить лишь уравнение плоскости, в которой этот маркер содержится. Все плоскости, получаемые от фиксированного детектора, пересекаются по одной прямой (“фокусная прямая”).

Задача состоит в том, чтобы по имеющимся данным в реальном времени восстанавливать положение твердого тела с максимальной точностью.

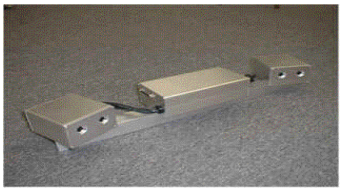


Рис. 1

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

На рис. 1 показан блок из четырех одномерных камер. Таких блоков, расположенных в разных частях пространства, может быть несколько.

ОСЛОЖНЯЮЩИЕ ФАКТОРЫ

1) В нескольких последовательных кадрах может быть недостаточно информации для однозначного восстановления положения тела (недоопределенность).
2) В силу несовершенства детектора возможна ситуация, когда он выдает ошибочный номер маркера.

МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА

Положение тела в пространстве можно представить в виде $Ax + b$, где $b = (u, v, w)$ — вектор сдвига, A — ортогональная матрица поворота тела относительно начального положения.

Задача будет решена, если будут найдены величины A, b, при которых отклонение положения маркеров от соответствующих плоскостей будет минимально в некоторой норме.

АЛГОРИТМ

Основная идея алгоритма заключается в том, чтобы в начале отфильтровать входные данные, а затем по ним восстановить положение тела. При этом недостающая информация о положении тела дополняется данными с предыдущих кадров на основе интерполяции.

1. Случай, когда на одном временном шаге данных (плоскостей) достаточно для однозначного восстановления положения тела

Матрица A задается направлением (p, q, r), относительно которого происходит поворот тела и синусом угла поворота $\sin\phi$. Заметим, что вектор (p, q, r) одновременно может задавать и угол поворота по формуле $p^2 + q^2 + r^2 = \sin^2\phi$ (мы используем модуль вектора для характеристики угла). Таким образом, матрица A задается тремя числами, и это представ-

ление есть непрерывное соответствие между ортогональными матрицами и единичным шаром в трехмерном пространстве. Все это позволяет нам для нахождения преобразования $Ax+b$ воспользоваться методом наименьших квадратов, который в случае достаточности данных позволяет сразу по исходным данным (уравнениям плоскостей) восстановить положение тела, минуя определение положения каждого маркера по отдельности (что зачастую невозможно, потому что на каждый маркер может приходиться менее чем три плоскости). Например, в случае, если на одном шаге времени видны одновременно 3 маркера, то для восстановления положения тела достаточно знать по две плоскости на каждый маркер (итого 6 плоскостей) или 1 плоскость про первый маркер, 2 плоскости про второй, 3 плоскости про третий маркер (также 6 плоскостей).

Метод наименьших квадратов позволяет также использовать избыточное количество плоскостей (более 6) для повышения точности восстановления.

Выпишем в явном виде [1] представление матрицы A:

$$A = A(p,q,r) = \begin{pmatrix} c & -r & q \\ r & c & -p \\ -q & p & c \end{pmatrix} + \frac{1}{1+c}\begin{pmatrix} p^2 & pq & pr \\ qp & q^2 & qr \\ rp & rq & r^2 \end{pmatrix},$$

где $c^2 = 1-(p^2+q^2+r^2) = \cos^2\phi$.

Пусть на данном временном шаге известны N плоскости, заданные в виде $(n_i, X) = d_i$, $i = 1\ldots N$, где $n_i = (a_i, b_i, c_i)$ — вектор нормали к плоскости, $(\circ,\circ)$ — скалярное произведение, а также N точки $X_i = (x_i, y_i, z_i)$, характеризующие координаты маркера в системе координат твердого тела (предполагается, что эта информация считывается из файла с описанием твердого тела).

Метод наименьших квадратов приводит к следующей задаче:

$$\sum_{i=1}^{N}\left[(n_i, AX_i + b) - d_i\right]^2 \to \min.$$

Взяв частные производные соответственно по p,q,r,u,v,w, приходим к следующей системе уравнений:

$$\begin{cases} \sum_{i=1}^{N}\left(\frac{\partial A}{\partial p}X_i, n_i\right)\left(\left(AX_i+b, n_i\right)-d_i\right)=0, \\ \sum_{i=1}^{N}\left(\frac{\partial A}{\partial p}X_i, n_i\right)\left(\left(AX_i+b, n_i\right)-d_i\right)=0, \\ \sum_{i=1}^{N}\left(\frac{\partial A}{\partial p}X_i, n_i\right)\left(\left(AX_i+b, n_i\right)-d_i\right)=0, \\ \sum_{i=1}^{N} a_i\left(\left(AX_i+b, n_i\right)-d_i\right)=0, \\ \sum_{i=1}^{N} b_i\left(\left(AX_i+b, n_i\right)-d_i\right)=0, \\ \sum_{i=1}^{N} c_i\left(\left(AX_i+b, n_i\right)-d_i\right)=0. \end{cases}$$

В полученную систему неизвестные u, v, w входят линейно, поэтому мы сразу можем их исключить (выразить через p, q, r) линейной заменой:

$$b = S^{-1}\sum_{i=1}^{N} d_i n_i - \sum_{i=1}^{N} S^{-1} S_i A X_i \text{, где}$$

$$S_i = \begin{pmatrix} a_i^2 & a_i b_i & a_i c_i \\ b_i a_i & b_i^2 & b_i c_i \\ c_i a_i & c_i b_i & c_i^2 \end{pmatrix},\ S = S_1 + S_2 + \ldots S_N .$$

В итоге у нас остается система трех нелинейных уравнений с тремя неизвестными p, q, r:

$$\begin{cases} f_1(p,q,r)=0, \\ f_2(p,q,r)=0, \\ f_3(p,q,r)=0, \end{cases}$$

или в векторном виде $F(P)=0$, где $F=(f_1, f_2, f_3)$, $P=(p,q,r)$.

Для ее решения воспользуемся методом последовательных приближений Ньютона [2].

Пусть на n-м шаге известно приближение $P_n=(p_n, q_n, r_n)$, тогда следующее приближение P_{n+1} находится по формуле: $P_{n+1}=P_n - J^{-1}F(P_n)$, где J — матрица Якоби преобразования F.

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial p} & \frac{\partial f_1}{\partial q} & \frac{\partial f_1}{\partial r} \\ \frac{\partial f_2}{\partial p} & \frac{\partial f_2}{\partial q} & \frac{\partial f_2}{\partial r} \\ \frac{\partial f_3}{\partial p} & \frac{\partial f_3}{\partial q} & \frac{\partial f_3}{\partial r} \end{pmatrix}.$$

Как известно, метод Ньютона очень быстро сходится в случае хорошего начального приближения.

В нашем алгоритме для начального приближения используются данные, полученные в результате экстраполяции по предыдущим временным шагам (с предыдущих кадров), — это обеспечивает хорошее начальное приближение и позволяет обойтись в методе Ньютона двумя-тремя итерациями для достижения нужной нам точности.

2. Случай, когда на одном временном шаге нам приходит недостаточное количество плоскостей (менее 6), для того чтобы однозначно восстановить положение тела

Очевидно, что в этом случае искомые величины p,q,r,u,v,w также удовлетворяют раннее описанной системе уравнений $F(P)=0$, но решений эта система имеет много (как правило, имеется целое пространство решений).

Недостающие плоскости мы получаем с помощью экстраполяции по предыдущим временным шагам следующим способом.

На первом этапе вычисляются положения невидимых в данный момент точек на основе того, что мы знаем их положение (следовательно, приближенно знаем вектор скорости и ускорения) на предыдущих шагах.

На втором этапе через вычисленные точки проводятся плоскости, которыми и дополняются исходные неполные данные.

Более точно, пусть (в таких же обозначениях, как и ранее) на данном временном шаге известны M дополнительных (полученных на основе интерполяции) плоскостей, заданных в виде $(n_i, X) = d_i$, $i = N+1 \ldots N+M$, а также M им соответствующих точек $X_i = (x_i, y_i, z_i)$.

Тогда в предположении, что $M + N \geq 6$ возникает задача

$$\sum_{i=1}^{N+M} \left[(n_i, AX_i + b) - d_i \right]^2 \to \min,$$

полностью аналогичная предыдущему случаю.

Заметим, что в принципе этим методом можно вычислять более точное положение тела на предыдущих временных шагах, используя информацию, полученную на последующих шагах (направить метод в “прошлое”).

3. Борьба с шумами

В силу несовершенства детекторов иногда фиксируются неверные плоскости (т.е. маркер не лежит в плоскости, а находится вдали от нее), например, если перепутается номер маркера.

Для борьбы с таким видом шумов положение тела восстанавливается дважды. Первый раз восстановление происходит по всем имеющимся плоскостям (в том числе и неверным).

Затем мы проверяем, насколько каждая плоскость отклоняется от вычисленного положения, и при втором восстановлении не учитываются те плоскости, в которых это отклонение велико (например, в 3 раза превосходит среднее отклонение и т.п.).

ОПИСАНИЕ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА

Была написана реализация [3] выше изложенного алгоритма. Язык C++ . В виде dll-библиотеки для Windows [6] и в виде запускаемого файла для Linux [7]. Программа тестировалась на данных, полученных с реальной системы, и показала высокую точность и устойчивость к шумам. Среднее время обработки 10000 временных шагов около 1 сек.

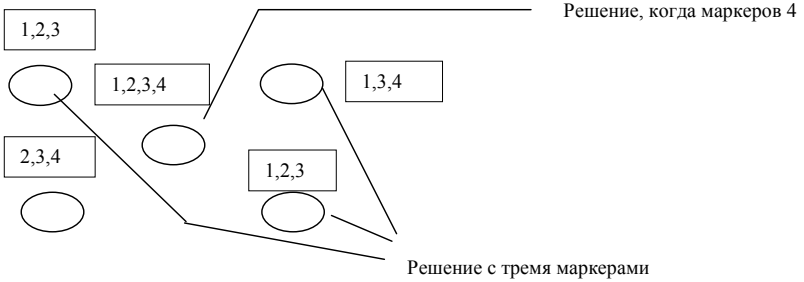


Рис. 2

Рис.2 поясняет эффект увеличения точности восстановления положения при использовании информации с большего количества детекторов.

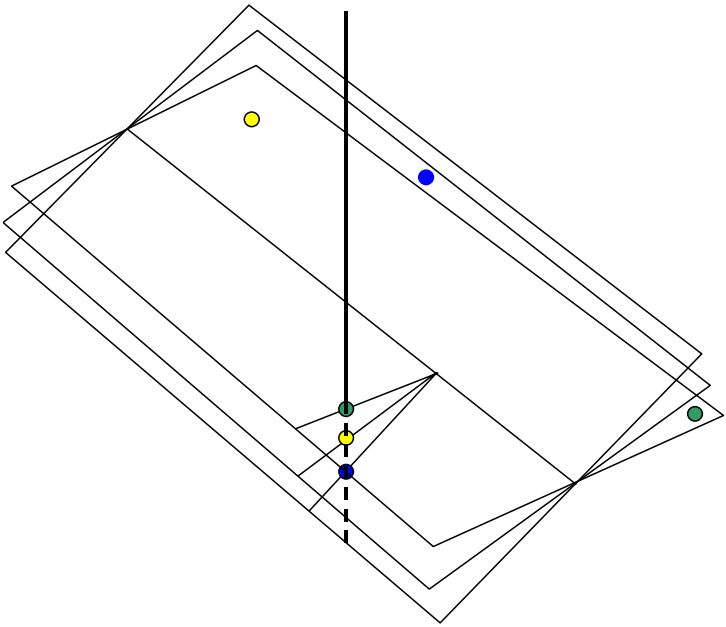


Рис. 3

На рис.3 жирной вертикальной линией показан детектор. Фокусная прямая — это есть пересечение трех плоскостей. Желтая, синяя и зеленая точки, не лежащие на детекторе, — это маркеры в пространстве. А точки на детекторе — соответствующие им пиксели.

СПИСОК ЛИТЕРАТУРЫ

1. **Александров П. С.** Лекции по аналитической геометрии. — М.: Наука, 1968. — С. 188
2. **Бахвалов Н.С., Жидков Н.П., Кобельков Г.М.** Численные методы. — М.: Наука, 1987.
3. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентированного проектирования. — СПб: Питер, 2001. — С.89–300.

4. **Pratt, William K.** Digital image processing. 2nd ed. — Wiley & Sons Ltd., 1991. — 438 p.
5. **The 4th All-Russian** with invited foreign participants Conf. “Pattern Recognition and Image Analysis: New Information Technologies”, ROAI-98, Novosibirsk: IA&E SibRAS, 1998. — Vol. 1, 2.
6. **Codeguru** http://www.codeguru.com
7. **Linux** http://www.linux.org/

Ф. А. Мурзин, Т. С. Мурзина, Е. М. Хаяров, В. Б. Шлишевский

СВЕТОСИЛЬНЫЕ РАСТРОВЫЕ СТРУКТУРЫ ДЛЯ РЕЖИМА АВТОКОЛЛИМАЦИИ*

В спектроскопии кроме щелевых спектрометров классического типа используются так называемые растровые спектрометры [1]. Например, в инфракрасной, а также в ультрафиолетовой областях, световые источники обычно бывают очень слабыми. Поэтому вместо щелей, используемых в щелевых спектрометрах, переходят к специальным растрам.

Использование растров позволяет в десятки и сотни раз увеличить выходящий световой поток при сохранении спектрального разрешения. Выходящий световой поток, собранный с площади, может быть сфокусирован на фотоприемник с помощью линзы, прозрачной в данном участке спектра, и таким образом преодолевается порог чувствительности датчика.

При конструировании такого рода приборов возникает целый ряд задач, относящихся к дискретной математике. Применяются методы алгебры, комбинаторики и теории чисел. Речь идет о поиске пар матриц больших размерностей (до 200.000), элементы которых +1 или –1, таких, что разного рода разностные корреляционные функции от них представляют собой дельта-функцию.

Например, мы сканируем два растра: позитив по позитиву и позитив по негативу. Требуется, чтобы разность получающихся корреляционных функций (сверток) была бы дельта-функцией, либо, в крайнем случае, дельта-функцией с небольшими осцилляциями [2,3]. При этом +1 соответствует черный квадрат на растре, –1 соответствует прозрачный квадрат, если растр изготовлен на прозрачной подложке (стекле или специальном кристалле), или соответствует отверстие в металлической пластине, что часто бывает для инфракрасных приборов.

В зависимости от типа сканирования имеются различные режимы: коммутация, осцилляция и автоколлимация. На практике, для прецизионных приборов наибольший интерес представляют два последних режима. Для режима осцилляции Т. С. Мурзиной и В. Б. Шлишевским [4] были предложены алгоритмы, позволяющие построить растровые структуры с разностными автокорреляционными функциями (РАКФ) вообще без побочных максимумов, на основе матриц Адамара. Растры такого вида внедрены в

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

серийные ИК-приборы. Для режима автоколлимации вопрос оставался открытым.

Были проведены теоретические исследования и численное моделирование, направленные на создание новых высокосветосильных растровых структур для режима автоколлимации.

В результате проведенных исследований предложен алгоритм построения высокосветосильных растровых структур для режима автоколлимации с разностными автокорреляционными функциями с очень малыми по амплитуде побочными максимумами. Для достижения этого результата используется метод, аналогичный предложенному Т. С. Мурзиной и В. Б. Шлишевским для режима осцилляции. Отметим, что получить чистую дельта-функцию вообще без побочных максимумов, как было получено для режима осцилляции, в данном случае, т.е. для режима автоколлимации, не удается. Ряд математических конструкций Т. С. Мурзиной и В. Б. Шлишевского в данном случае не работают, и их методы можно применить лишь частично. Численные эксперименты выполнялись в системе Maple_V_R4.

СПИСОК ЛИТЕРАТУРЫ

1. **Бухонин В.С., Чиков К.Н., Шлишевский В.Б.** Светосильная растровая спектроскопия // Новые методы спектроскопии. — Новосибирск: Наука, 1982. — С. 3–77.
2. **Murzin F.A., Murzina T.S., Shlishevsky V.B.** New Grills for Girard Spectrometers // Applied Optics. — 1985. — V. 24, N 21. — P. 3625–3630**.**
3. **Мурзин Ф.А., Мурзина Т.С., Чайка Н.Ф., Шлишевский В.Б.** Светосильная растровая спектроскопия на основе матриц Адамара. — Новосибирск, 1984. — 51 с. — (Препр / Сиб. отд-ние РАН. ИТПМ, НИИГАиК, Зап.Сиб.РНИИ; №17-84).
4. **Мурзина Т.С., Шлишевский В.Б.** Исследование растровых систем на основе матриц Адамара, Мат. модели в геодезии, кадастре и оптотехнике // Сибирская государственная геодезическая академия. — Новосибирск, 1999. — С. 52–58.

Ф. А. Мурзин, О. Н. Половинко, И. В. Лобив

РАСПОЗНАВАНИЕ ТЕКСТУР ПО ПРОСТРАНСТВЕННЫМ ЗАКОНОМЕРНОСТЯМ*

ВВЕДЕНИЕ

Статистический и структурный подходы к описанию текстур и оптические методы реализации этих подходов достаточно полно раскрыты в работе Р. М. Харалика [3].

В данной статье описываются рассмотренные нами варианты применения данных способов при цифровой обработке изображений и приводятся полученные результаты.

Известно восемь статистических подходов к измерению и описанию текстурных характеристик изображения, нами проводилась работа с использованием автокорреляционных функций, плотности перепадов и длин серий.

Основной сложностью при изучении свойств текстур (под понятием текстуры мы понимаем некоторым образом организованный участок поверхности), является тот факт, что очень сложно разработать универсальный метод распознавания. Иными словами можно сказать, что под любой вид текстуры можно подобрать метод распознавания, который, при качественной настройке будет выдавать практически стопроцентный результат, тогда как на другом виде текстуры этот метод работать не будет.

Для демонстрации практических результатов выбраны представители наиболее часто встречаемых — естественных, структурных и стохастических — видов текстур.

Написана программа на языке программирования Visual C++ 6.0, реализующая три подхода к изучению текстурных характеристик. В программе используется система цветовых координат RGB. Цветное изображение размером $n \times m$ задается тремя матрицами $I_R = I_R(i, j)$, $I_G = I_G(i, j)$ и $I_B = I_B(i, j)$, где $0 \le i \le n-1$, $0 \le j \le m-1$. Значения элементов матриц $I_R(i, j)$, $I_G(i, j)$ и $I_B(i, j)$ изменяются в пределах от 0 до 255.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

Все тесты проводились на компьютере с процессором Pentium III (700 MHz).

РЕАЛИЗОВАННЫЕ МЕТОДЫ

1. Плотность перепадов

1.1. Теоретическая часть

Впервые описывать текстуру, исходя из количества перепадов яркости на единицу площади изображения, попробовали Розенфельд и Трой, а также Розенфельд и Терстон. Идеи последних были развиты Саттоном и Холлом, которые достигли 80%-ой точности в различении больных и здоровых легких при размере изображения 128 * 128.

Итак, при данном подходе текстуру трактуют, исходя из количества перепадов яркости на единицу площади изображения. Приходящийся на клетку изображения перепад можно обнаружить, сравнивая значения локальных признаков пар непересекающихся фрагментов изображения, соседствующих с этой клеткой.

В качестве локального признака Розенфельд и Терстон предложили использовать сокращенный градиент Робертса (сумму абсолютных значений разностей между уровнями яркости пар соседних клеток, расположенных по обе стороны от каждой диагонали).

Саттон и Холл предложили вычислять функциональную зависимость градиента от расстояния между точечными элементами. На множестве соседних точек $\boldsymbol{N}$ и в указанных выше обозначениях эта функция записывается в виде:

$$g(d) = \sum_{(i,j)\in N} \{ | I(i,j) - I(i+d,j) | + | I(i,j) - I(i-d,j) | + | I(i,j) - I(i,j-d) | + | I(i,j) + I(i,j+d) | \},$$

где d — расстояние. График функции $g(d)$ напоминает смещенный вверх график автокорреляционной функции, взятой со знаком минус.

1.2. Проделанная работа

Данный подход реализован функцией, в основу которой положен следующий структурный алгоритм (рис.1).

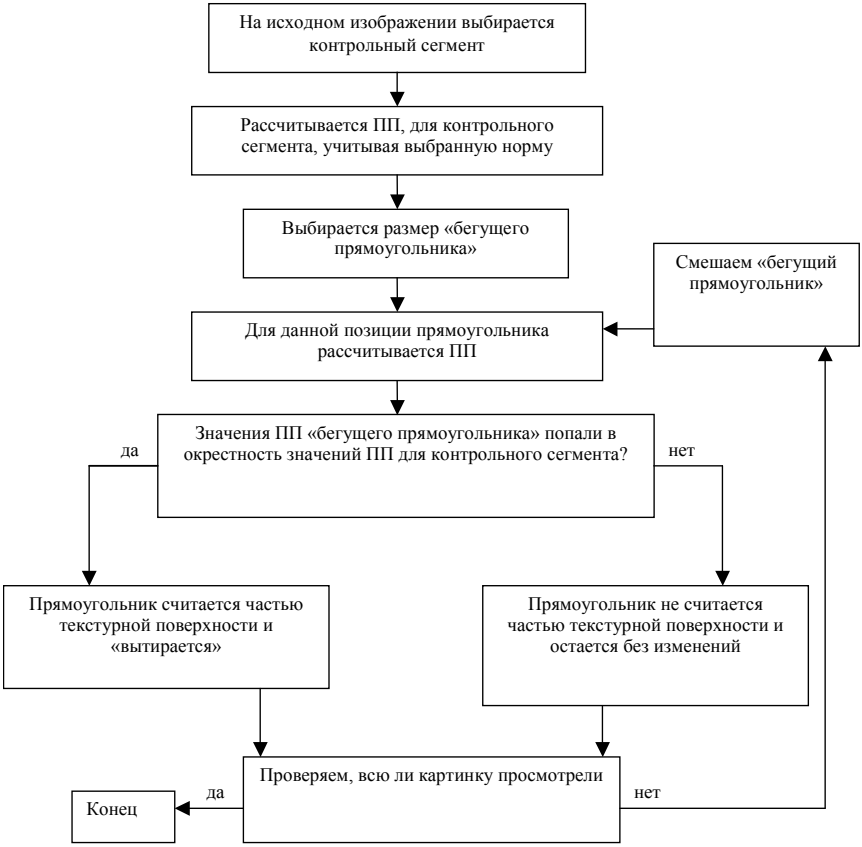


Рис. 1. Блок-схема алгоритма функции ПП

При вычислении частоты перепадов применялось несколько норм для определения близости цветов. Опыты показали, что оптимальные результаты при определении близости цветов достигаются при применении шаровой нормы, т.е. суммы квадратов значений яркости, с радиусом равным ста, т.е.

$$R^2 + G^2 + B^2 = C = 100.$$

Величина коэффициента определялась способом визуального восприятия, т.е. провели несколько опытов, показывающих, какие оттенки цветов

различимы для человеческого глаза, и была взята минимальная величина по значениям, полученным в проведенных опытах.

Для сравнения величин частот был применен следующий метод, бралась их разность по модулю и сравнивалась с некоторой погрешностью. О величине погрешности можно сказать следующее: величины для разных тестовых изображений становились различными, начиная с 0.15 и больше. Кроме того, о погрешности можно сказать то, что она является функцией от размера самого изображения, от размера контрольного сегмента, и от размера «бегущий прямоугольник».

1.3. Полученные результаты

Приведем примеры обработки этим методом тестовых изображений.

Тест №1. «Кирпичная стена» (рис.2).

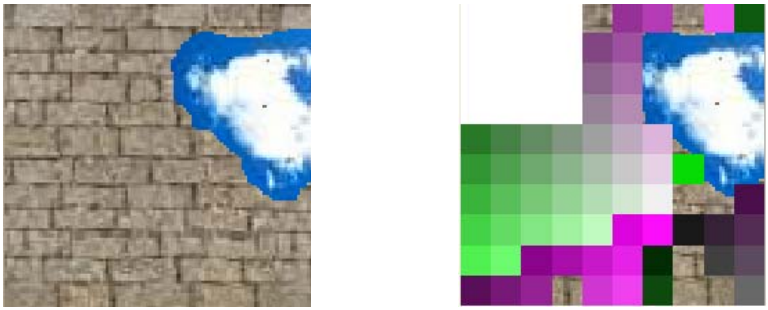


Рис.2. Тест №1. «Кирпичная стена», обработка методом ПП

Параметры:

величина картинки 100×100 ,
величина контр. сегмента 40×40 ,
величина константы $C = 100$,
точность обработки 92%,
скорость обработки 0.01 сек,
величина «Бег. Пр.» 10×10 ,
величина погрешности 0.2.

Тест №2. «Пузырьки» (рис. 3).

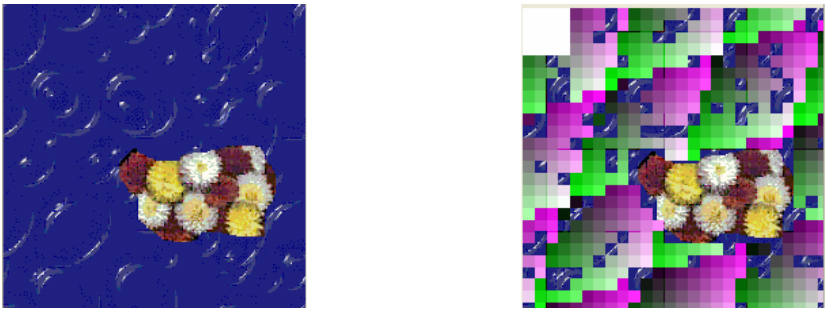


Рис.3. Тест №2. «Пузырьки», обработка методом ПП

Параметры:

величина картинки 250×250 ,
точность обработки 90%,
величина контр. сегмента 40×40 ,
величина «Бег. Пр.» 10×10 ,
величина константы $C = 100$,
величина погрешности 0.2,
скорость обработки 0.03 сек.

Тест №3. «Полотенце» (рис. 4).

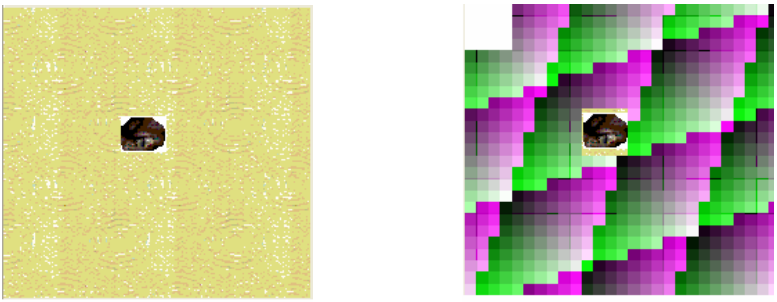


Рис. 4. Тест №3. «Полотенце», обработка методом ПП

Параметры:

величина картинки 265×250 ,
точность обработки 100%,

величина контр. сегмента 40×40 ,
величина «Бег. Пр.» 10×10 ,
величина константы $C = 100$,
величина погрешности 0.2,
скорость обработки 0.03 сек.

2. Автокорреляционная функция

2.1. Теоретическая часть

В этом подходе текстура связана с пространственным размером тоновых непроизводных элементов изображения (*тоновый непроизводный элемент* — это область изображения с определенными тоновыми признаками). Значение автокорреляционной функции является как раз тем признаком, который характеризует размер тоновых непроизводных элементов. Пространственное расположение характеризуется коэффициентом корреляции, который является мерой линейной зависимости яркости одного элемента изображения от яркости другого [3].

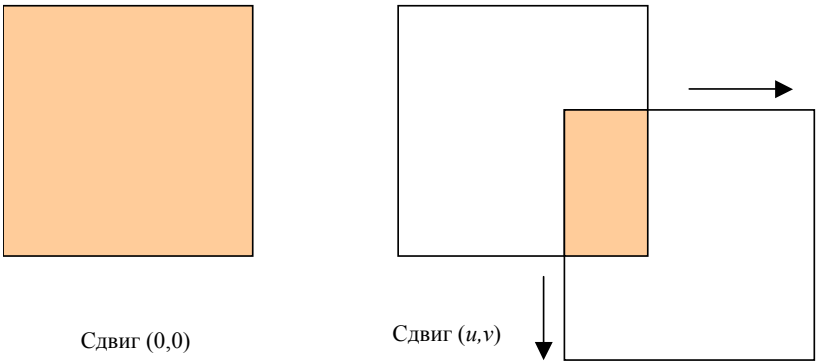


Рис.5 Метод нахождения значений АКФ

Пояснить идею работы этого метода можно на следующем примере (рис. 5). Рассмотрим два одинаковых диапозитива одного и того же изображения. Наложим один на другой, осветим однородным пучком света и измерим среднюю величину светового потока, прошедшего через такой двойной диапозитив. После этого сдвинем один диапозитив относительно другого и замерим средний световой поток только в той части изображения, в которой диапозитивы перекрываются. Эта величина, поделенная на средний световой поток при нулевом (*0,0*) сдвиге, образует значение двумерной

автокорреляционной функции диапозитива, зависящее от координат (x,y) сдвига.

Математически процесс расчета автокорреляционной функции можно описать следующей формулой:

$$\rho(x,y)=\frac{\frac{1}{(L_x-|x|)\times(L_y-|y|)}\iint S(u,v)\times S(u+x,v+y)\,dudv}{\frac{1}{L_x\times L_y}\iint S^2(u.v)\,dudv},$$

где $S(u,v)$ — прозрачность диапозитива в точке с координатами (u,v), (x,y) — величина сдвига в направлении x и y. Также предполагаем, что вне прямоугольника $0\le u\le L_x$, $0\le v\le L_y$ прозрачность равна нулю.

2.2. Проделанная работа

В нашем случае, рассматривалось не черно-белое, а цветное изображение и роль прозрачности выполняла яркость или цветовая насыщенность. Кроме того, так как функция дискретна, то берется не отношение интегралов, а отношение частичных сумм. А для каждой компоненты цвета строится своя автокорреляционная функция.

$$\rho_R(x,y)=\frac{\frac{1}{(L_x-|x|)\times(L_y-|y|)}\sum\sum I_R(i,j)\times I_R(i+x,j+y)}{\frac{1}{L_x\times L_y}\sum\sum I_R^2(i,j)};$$

$$\rho_G(x,y)=\frac{\frac{1}{(L_x-|x|)\times(L_y-|y|)}\sum\sum I_G(i,j)\times I_G(i+x,j+y)}{\frac{1}{L_x\times L_y}\sum\sum I_G^2(i,j)};$$

$$\rho_B(x,y)=\frac{\frac{1}{(L_x-|x|)\times(L_y-|y|)}\sum\sum I_B(i,j)\times I_B(i+x,j+y)}{\frac{1}{L_x\times L_y}\sum\sum I_B^2(i,j)}.$$

Структура алгоритма вычисления автокорреляционной функции схожа со структурой алгоритма вычисления плотности перепадов. Здесь выполняется расчет значений функции для контрольного сегмента, а затем происхо-

дит «пробежка» по всему изображению. Причем размер контрольного сегмента больше либо равен размеру «бегущего прямоугольника».

При сравнении значений, получаемых при «пробежке» с контрольным значением, пришлось учитывать возможный сдвиг фаз и некоторую погрешность в определении близости значений автокорреляционных функций.

Кроме того, была написана функция на OpenGL, которая выводит график автокорреляционной функции контрольного сегмента.

2.3. Полученные результаты

Приведем примеры обработки этим методом тестовых изображений.

Тест №4. «Кирпичная стена» (рис.6).

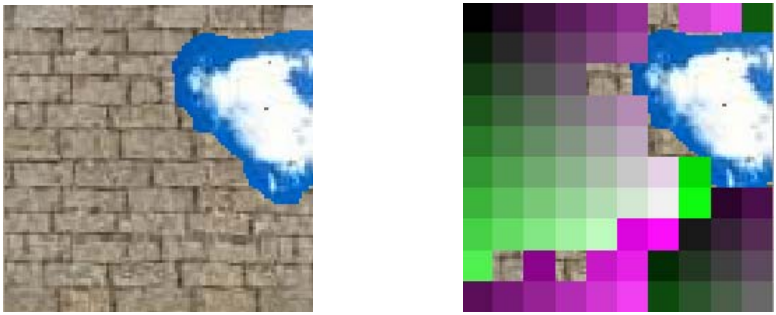


Рис. 6. Тест №4. «Кирпичная стена», обработка методом АКФ

Графики автокорреляционных функций для данного теста выглядят следующим образом (рис.7).

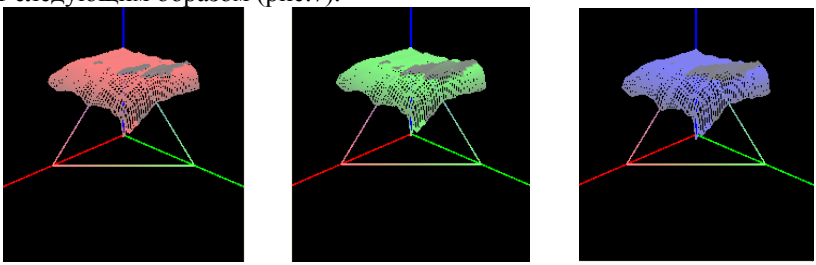


Рис. 7. Графики АКФ для теста №4

Параметры:

величина картинки 100×100 ,
точность обработки 96%,
величина контр. сегмента 40×40 ,
величина «Бег. Пр.» 10×10 ,
величина погрешности 0.1,
скорость обработки 0.2 сек.

Тест №5. «Пузырьки» (рис. 8).

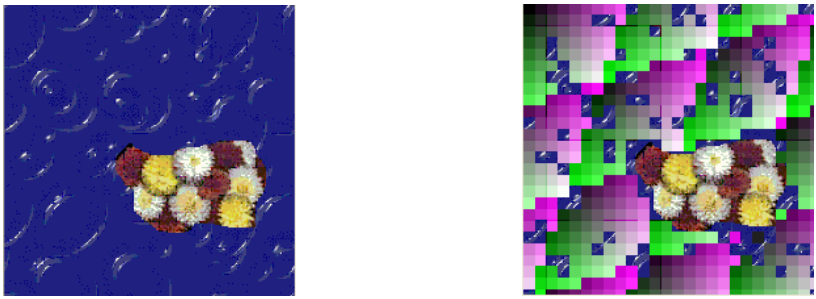


Рис. 8. Тест №5. «Пузырьки», обработка методом АКФ

Графики АКФ для данного теста (рис.9).

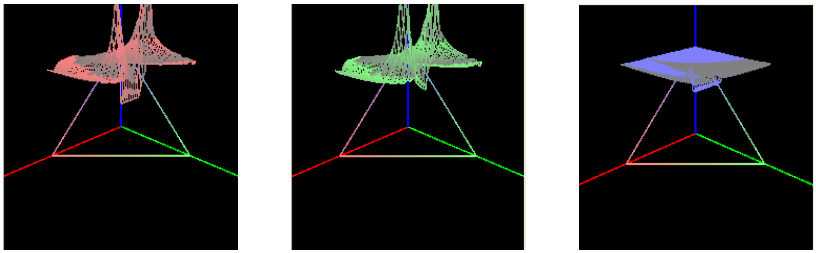


Рис. 9. Графики АКФ для теста № 5

Параметры:

величина картинки 250×250 ,
точность обработки 83%,
величина конт. сегмента 40×40 ,
величина «Бег. Пр.» 10×10 ,
величина погрешности 0.1,
скорость обработки 0.381 сек.

Тест №6. «Полотенце» (рис.10).

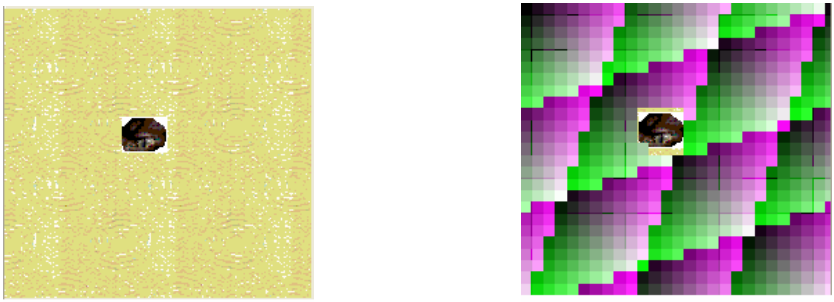


Рис. 10. Тест №6. «Полотенце», обработка методом АКФ

Графики АКФ для данного теста (рис.11).

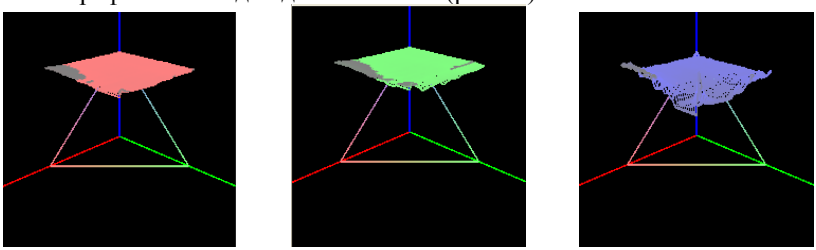


Рис.11. Графики АКФ для теста №6

Параметры:

величина картинки 265×250 ,
точность обработки 100%,
величина контр. сегмента 40×40 ,
величина «Бег. Пр.» 10×10 ,
величина погрешности 0.1,
скорость обработки 0.24 сек.

3. Длины серий

3.1. Теоретическая часть

Серией называется непроизводный элемент, состоящий из максимальной связной совокупности вытянутых в прямую линию точечных элементов изображения одинаковой яркости. Серия характеризуется яркостью, длиной и направлением [3]. В проделанной работе оценивалась совместная вероятность яркости и длины серии в каждом из четырех направлений: 0^0, 45^0, 90^0, 135^0.

При распознавании использовались некоторые статистики величины, $p(i, j)$ — число серий длины j с яркостью i, например, такие как моменты, неоднородность яркости, неоднородность длин серий, доля изображения в сериях. Пусть N_g — число возможных значений яркости, N_r — число возможных длин серий, тогда математически эти характеристики выражаются следующими формулами:

$\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{p(i,j)}{j^2}\Big/\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}p(i,j)$ — обратные моменты увеличены при коротких сериях,

$\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}j^2 p(i,j)\Big/\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}p(i,j)$ — моменты увеличены при длинных сериях,

$\sum_{i=1}^{N_g}\left(\sum_{j=1}^{N_r}p(i,j)\right)^2\Big/\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}p(i,j)$ — неоднородность яркости,

$\sum_{j=1}^{N_r}\left(\sum_{i=1}^{N_g}p(i,j)\right)^2\Big/\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}p(i,j)$ — неоднородность длины серии,

$\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}p(i,j)\Big/\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}jp(i,j)$ — доля изображения в сериях.

3.2. Проделанная работа

При реализации данного подхода к выявлению текстурных признаков возникла идея рассматривать серии, характеризуемые цветовой насыщенностью, т.е. накапливать информацию о сериях красного, синего и зеленого цветов. Кроме того, был реализован подход, в котором серия характеризу-

ется просто цветом. Для определения близости цветов использовалась та же норма, что и в плотности перепадов.

На данном этапе, идет работа над способами сравнения характеристик серий. Реализован способ сравнения одинаковых характеристик по соответствующим направлениям.

На данный момент ничего нельзя сказать о реальных возможностях данного метода, так как он требует дальнейшей доработки и анализа. Кроме того, не набрана достаточная статистика, для того чтобы подобрать оптимальные значения для используемых параметров.

ЗАКЛЮЧЕНИЕ

В работе исследовалась задача распознавания текстуры по некоторому ее участку, а точнее проводился анализ закономерностей характеристик текстурного изображения.

Рассмотрены три метода, основанные на анализе плотностей перепадов, автокорреляционных функций и различных статистических характеристик серий.

В методе, использующем анализ плотностей перепадов, была достигнута 90—100% точность вытирания текстуры. Такой результат был получен при применении шаровой нормы для определения близости цветов, радиус нормы равен ста.

При использовании сравнений автокорреляционных функций точность вытирания текстуры составила 83 — 100%. Применялась та же норма, что и в анализе плотностей перепадов. При непосредственном сравнении автокорреляционных функций был учтен возможный сдвиг фаз. Опыты показали, что автокорреляционная функция периодически убывает и возрастает в соответствии с мерой периодичности пространственного расположения тоновых непроизводных элементов.

На данном этапе работы над выявлением закономерностей характеристик серии уже реализован метод сравнения соответствующих характеристик участков изображения. Но такой подход дает очень слабый результат. Сейчас ведется работа по разработке лучшего способа сравнения характеристик серий, поэтому о результатах применения этого метода говорить еще рано.

СПИСОК ЛИТЕРАТУРЫ

1. **Мурзина Т.С., Шлишевский В.Б.** Растровые системы на основе матриц Адамара для растровых спектрометров. Математические результаты. — Новосибирск, 1991. — (Препр. / ВЦ СОРАН, 1991).
2. **Прэтт У.К., Фожра О.Д., Гагалович А.** Применение моделей стохастических текстур для обработки изображений // ТИИЭР. — 1981. — Т. 69, № 5.
3. **Харалик Р. М.** Статистический и структурный подходы к описанию текстур // ТИИЭР. — 1979. — Т. 67, № 5.
4. **Дунаев А. А., Лобив И. В., Мехонцев Д. Ю., Мурзин Ф. А., Половинко О. Н., Семич Д. Ф., Чепель А. В., Ярков К. А.** Алгоритмы быстрого поиска фрагментов фотографических изображений // Современные проблемы конструирования программ. — Новосибирск, 2002. — С. 88 — 110.

Д. Ю. Мухин

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ВАРЬИРОВАНИЯ ПРИМЕНИТЕЛЬНО К MIDI-ФАЙЛАМ*

ВВЕДЕНИЕ

Одним из существенных элементов процесса творчества является процедура транспонирования (переноса) некоторой структуры из одной ситуации в другую. Мы займемся исследованием закономерностей таких переносов и их места в общем процессе творчества, также изучением методов формирования возникающих при этом вариантов и способов их оценки.

Известно, что восприятие характеризуется такой качественной особенностью, как структурность. Структура, целостная и неразложенная на составляющие части, носитель некоторого образа. Возможность перенесения структуры из одной ситуации в другую, в новые условия, или транспонируемость, является основным свойством структуры.

В нашем случае структурой будет являться мелодия, формализованная в виде символьной последовательности. Каждой мелодии мы поставим в соответствие три числовых последовательности.

1. Последовательность номеров нот. Все ноты, начиная от ноты «до» первой октавы, мы пронумеровываем натуральными числами от 0 до 128.

2. Последовательность интервалов. Последовательность звуковысотных интервалов между соседними нотами. Если каждый интервал выразить числом полутонов, то такую последовательность можно записать в виде цепочки целых чисел (знак «-» будет показывать движение текущей ноты вниз относительно непосредственно предшествующей ей ноты, отсутствие знака перед интервалом — движение текущей ноты вверх).

3. **Последовательность длительностей.**

Одним из методов изучения мышления является варьирование ситуаций. При изменении некоторой первоначальной ситуации получается варьиро-

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

ванная ситуация, но при восприятии исходной и варьированной ситуаций обнаруживается их известная общность.

Наряду с новым в варьированной ситуации ощущается непосредственная связь с первоначальной ситуацией, элемент повторяемости или постоянства. Это обстоятельство связано с наличием инвариантов преобразования — таких элементов ситуации, которые остаются неизменными, постоянными при любой трансформации. Иначе говоря, при варьировании происходит перенос определенных отношений элементов из одной ситуации в другую, т.е. сохранение некоторых исходных структурных свойств.

Наличие инвариантов в варьированной и исходной ситуациях часто маскируется другими элементами, которые как бы меняют лицо первоначальной ситуации. Это трансформанты — элементы, изменяющиеся при преобразовании. Маскирующие элементы часто до неузнаваемости трансформируют ситуацию и весьма затрудняют обнаружение инвариантов преобразования.

Все трансформации можно разделить на три основных класса.

1. Трансформации звуковысотности. При таких трансформациях осуществляются изменения в последовательности высот звуков. Например, элементарной трансформацией такого типа будет перенос всей мелодии по вертикали. В рамках нашей модели — увеличение каждого из чисел (которыми закодирована каждая нота нашей мелодии) на фиксированное число n.

2. Трансформации ритма. При таких трансформациях изменения вносятся в ритмическую часть мелодии. Например, изменение длительностей нот или метрических акцентов. В рамках нашей модели: изменяться будут числа, которыми закодированы длительности нот.

3. Орнаментальные трансформации. При трансформациях такого типа изменения вносятся как в звуковысотную, так и в ритмическую часть мелодии. Примером будет являться такая комбинация трансформаций: каждая длительность ноты дробится на несколько частей (одну, две, три, четыре), в каждом дроблении высота первой ноты совпадает с высотой ноты в теме до дробления.

При прослушивании примеров к каждому классу трансформаций, мы заметили, что в некоторых образах искомая мелодия угадывалась сразу, а в некоторых была изменена до неузнаваемости. Секрет в том, что эти разные последовательности звуков объединяет какое-то одно из их общих качеств.

В примере для первого класса трансформаций была одна и та же последовательность звуковысотных интервалов между соседними нотами. Здесь именно эта последовательность интервалов и способствует восприятию одной и той же мелодии при прослушивании разных последовательностей нот. Однако та инвариантная структура, которая служит носителем образа мелодии, не сводится к одной лишь неизменной последовательности интервалов. Такая структура в нашем первом примере состоит из двух инвариантов — последовательности интервалов и ритма. Если же разложить ее на составляющие части — ритм и последовательность интервалов, то взятая в отдельности каждая часть, являясь инвариантом, тем не менее, может и не быть носителем первоначального образа.

В примерах для второго класса трансформаций в первом случае инвариантами являлись последовательность высот звуков и ритм, а во втором — только ритм.

В примерах для третьего класса трансформаций инвариантом была неизменная последовательность высот первых нот каждой четверки с сохранением метрических акцентов.

Итак, тот или другой инвариант обеспечивает сохранение первоначального образа — образа исходной ситуации при варьировании. Для этого необходима инвариантная структура — совокупность некоторых инвариантов, которая и является носителем данного образа.

Таким образом, инвариантная структура определяется не элементами, ее образующими, а системой отношений между этими элементами. При транспонировании осуществляется изменение элементов, звуков, составляющих мелодию, по высоте и другим параметрам, но сохраняется система отношений между ними.

1. ПОСТАНОВКА ЗАДАЧИ

На входе имеется несколько MIDI-файлов, на первой дорожке каждого из которых записана одноголосая мелодия. Один из файлов — эталон (исходная мелодия), остальные — эталон, трансформированный разными способами. Наша задача заключается в том, чтобы установить степень сходства между трансформированными мелодиями и мелодией-эталоном и проанализировать изменение степени в зависимости от произведенных трансформаций.

Далее, получаем результаты для MIDI-файлов, каждый из которых конвертируется в WAVE-файл и проводим уже спектральный анализ (основным объектом изучения уже будет звуковой сигнал).

Результаты обоих типов сравниваются.

2. ЗАДАЧА НЕЧЕТКОГО СОПОСТАВЛЕНИЯ СТРОК

Строка длины $|x|=m$ записывается как $x_1x_2...x_m$, где x_i представляет i-й символ x. Подстрока $x_ix_{i+1}...x_j$ строки x, где $i \le j \le m$, будет обозначаться $x(i,j)$.

Обобщенная задача сопоставления строк, включающая в себя нахождение подстрок строки текста, близких к заданному образцу строки, называется также задачей нечеткого сопоставления строк. Задачу нечеткого сопоставления строк можно сформулировать следующим образом.

Пусть даны образец x, $|x|=m$, и текст y, $|y|=n$. Пусть даны также целое $k>0$ и функция расстояния d. Требуется найти все подстроки s текста y такие, что $d(x,s)<k$. Здесь и далее d — метрика.

Расстояние Хемминга [Hamming, 1982] между двумя строками одинаковой длины определяется как число позиций, в которых символы не совпадают. Это эквивалентно минимальной цене преобразования первой строки во вторую в случае, когда разрешена только операция замены с единичным весом. Если допускается сравнение строк разной длины, то как правило, требуются также вставка и удаление. Если придать им тот же вес, что и замене, минимальная общая цена преобразования будет равна одной из метрик, предложенных Левенштейном [Levenstein, 1965]. Нас будет интересовать метрика Левенштейна, т.к. в случае трансформации ритмической части мелодии длины наших числовых последовательностей могут не совпадать.

Задача, таким образом, состоит в том, чтобы при заданной функции расстояния найти все подстроки текста, отстоящие от образца не более чем на k. Если d является расстоянием Хемминга, задача называется сопоставлением строк с k несовпадениями, если же d — расстояние Левенштейна, задача называется сопоставлением строк с различиями (или иногда ошибками).

2.1. Алгоритм Вагнера—Фишера

В методе динамического программирования последовательно по предыдущим значениям вычисляются расстояния между все более и более длинными префиксами двух строк — до получения окончательного результата. Опишем этот процесс более подробно.

Пусть $d_{i,j}$ есть расстояние между префиксами строк x и y, длины которых равны соответственно i и j, т. е. $d_{i,j} = d(x(1,i), y(1,j))$.

Цену преобразования символа a в b обозначим через $w(a,b)$. Таким образом, $w(a,b)$ — это цена замены одного символа на другой, когда $a \neq b$, $w(a,\varepsilon)$ — цена удаления a, а $w(\varepsilon,b)$ — цена вставки b. Заметим, что в случае, когда выполнены нижеследующие условия, d является расстоянием Левенштейна

$$w(a,\varepsilon) = 1,$$
$$w(\varepsilon,b) = 1,$$
$$w(a,b) = 1, \text{ если } a \neq b,$$
$$w(a,b) = 1, \text{ если } a = b.$$

В процессе вычислений значения $d_{i,j}$ записываются в массив $(m+1)\times(n+1)$, а вычисляются они с помощью следующего рекуррентного соотношения:

$$d_{i,j} = \min\{d_{i-1,j} + w(x_i,\varepsilon), d_{i,j-1} + w(\varepsilon,y_j), d_{i-1,j-1} + w(x_i,y_i)\}.$$

Оно выводится следующим образом. Если предположить, что известна цена преобразования $x(1, i-1)$ в $y(1, j)$, то цену преобразования $x(1,i)$ в $y(1, j)$ мы получим, добавив к ней цену удаления x_i. Аналогично, цену преобразования $x(1,i)$ в $y(1, j)$ можно получить, прибавив цену вставки y_j к цене преобразования $x(1,i)$ в $y(1, j-1)$. Наконец, зная цену преобразования $x(1, i-1)$ в $y(1, j-1)$, цену преобразования $x(1, i)$ в $y(1, j)$ мы получим, прибавив к ней цену замены x_i на y_j. Вспомним, что расстояние $d_{i,j}$ является минимальной ценой преобразования $x(1, i)$ в $y(1, j)$, поэтому из трех указанных выше операций надо выбрать самую дешевую.

Перед тем как начать вычислять $d_{i,j}$, надо установить граничные значения массива. Что касается первого столбца массива, то значение $d_{i,0}$ равно сумме цен удаления первых i символов x. Аналогично, значения $d_{0,j}$ первой строки задаются суммой цен вставки первых j символов y. Итак, имеем следующее:

$$d_{0,0} = 0,$$

$$d_{i,0} = \sum_{k=1}^{i} w(x_i, \varepsilon) \text{ для } 1 \le i \le m,$$

$$d_{0,j} = \sum_{k=1}^{j} w(\varepsilon, j_k) \text{ для } 1 \le j \le n.$$

Для расстояния Левенштейна $d_{i,0} = i$ и $d_{0,j} = j$. Ниже приведен массив, полученный при вычислении расстояния Левенштейна, например, между строками «12345 » и «12354 ». Из него видно, что расстояние между этими строками, т.е. $d_{5,5}$, равно 2:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 1 & 2 & 3 & 4 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 4 & 3 & 2 & 1 & 1 & 1 \\ 5 & 4 & 3 & 2 & 1 & 2 \end{pmatrix}.$$

Алгоритм вычисления массива расстояний разработан Вагнером и Фишером (Wagner, Fisher). Можно видеть, что стадия инициализации границ включает $1+m+n$, а основной цикл повторяется mn раз. Таким образом, временная сложность этого алгоритма есть $O(mn)$. Такое положение вещей нас устраивает, так как мы имеем дело с числовыми последовательностями сравнительно малой длины.

2.2. Анализ результатов (MIDI)

Эталон (обозначим его 0-й мелодией) — MIDI-файл с одноголосой мелодией из известного классического произведения. В таблице приведены четыре трансформации звуковысотности — мелодия была сыграна в четырех различных гаммах (назовем трансформированные мелодии соответст-

венно 1-й, 2-й, 3-й и 4-й). Заранее была субъективно установлена степень сходства между эталоном и трансформированными мелодиями.

При сравнении эталона и 1-ой мелодии на слух образ сохранился полностью, при сравнении эталона и последующих мелодий субъективно при прослушивании сходство распределилось по убывающей.

Далее приводятся результаты сравнения с помощью метрики Левенштейна (L):

Сравниваемые мелодии	L для посл-ти нот	L для посл-ти интервалов	L для посл-ти длительностей
0 & 1	24	0	0
0 & 2	5	10	0
0 & 3	10	16	0
0 & 4	13	20	0

В первом случае была применена трансформация звуковысотности — перенос всей мелодии по вертикали на пять нот вверх. Несмотря на то что дистанция для последовательности нот принимает наибольшее значение из всех, образ сохранился полностью. Это произошло из-за наличия инварианта — последовательности интервалов, на что указывает нулевая дистанция для последовательности интервалов.

В остальных трех случаях производились более сложные трансформации звуковысотности, для этого использовалась программа *Ntonyx Style Morpher*. В результате получилось, что меньшему изменению образа соответствуют меньшие значения метрики Левенштейна (L) для последовательностей нот, для последовательностей интервалов и для последовательностей длительностей.

3. ПРЕОБРАЗОВАНИЕ АДАМАРА

Теперь, сконвертировав проанализированные MIDI-файлы в формат WAVE, подойдем к задаче исследования алгоритмов варьирования с точки зрения спектрального анализа звуковых сигналов.

Существует бесконечное количество ортонормированных базисов, по которым можно раскладывать сигналы. Сейчас самым распространённым и наиболее естественным является базис Фурье тригонометрических функций

(sin и cos). Но в некоторых ситуациях другие базисы оказываются более оптимальными с точки зрения скорости выполнения преобразования на компьютере, вида коэффициентов и т.д. Может оказаться, что для данного вида сигналов больше подходит преобразование, отличное от Фурье. Поэтому при обработке сигналов желательно иметь достаточно широкий набор различных преобразований для более глубокого анализа. Одним из таких преобразований и является преобразование Уолша-Адамара. Его отличительной особенностью является то, что сигнал раскладывается не по синусам и косинусам, а по функциям Уолша, принимающим значения ±1, что значительно облегчает вычисления на компьютере и позволяет быстро оперировать большими объёмами информации. Для цифровой обработки звуковых сигналов необходимо представить их в цифровом виде, т.е. провести аналого-цифровое преобразование, которое заключается в дискретизации (разбиения области определения) и квантовании (отображении области значений в дискретный диапазон). Амплитуда кодируется целым числом в диапазоне от –32768 до 32767. По теореме Найквиста-Котельникова максимальная частота равна ½ частоты дискретизации. Стандартная дискретизация CD-качества равна 44100Hz, следовательно максимальная воспроизводимая частота — 22050Hz, что не намного превышает максимальную частоту, слышимую человеком. Таким образом, сигнал в цифровом представлении — это последовательность чисел, которую мы и будем преобразовывать с помощью дискретного преобразования Уолша-Адамара.

Рассмотрим матрицу Адамара, которая строится следующим способом:

$$H_4 = \begin{bmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & \begin{pmatrix} -1 & -1 \\ -1 & 1 \end{pmatrix} \end{bmatrix}, \quad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{2^{n-1}} = \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix}.$$

В этой матрице необходимо упорядочить строки по числу перемен знаков, или, как говорят, упорядочить по Уолшу:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}.$$

Каждой базисной функции Уолша, или базисному сигналу, сопоставляется строка в матрице Адамара, причём число перемен знаков — это аналог частоты этого сигнала.

Собственно преобразование заключается в том, что матрица Адамара размерности $2n$ умножается на столбец из $2n$ значений, соответствующих дискретизированному сигналу, в результате чего мы получаем вектор, компонентами которого являются коэффициенты разложения данного дискретного сигнала по дискретным функциям Уолша. В данном случае $2n$ — это количество временных отсчётов в соответствии с заданной дискретизацией.

Как определить частоту волны? Приближённая формула получается следующим образом. Берём коэффициент с наибольшим модулем и составляем пропорцию из условия, что 65536-ый коэффициент соответствует частоте 22050Hz:

$$\upsilon = \frac{k \cdot \upsilon_{\max}}{k_{\max}}.$$

Здесь υ — искомая частота, k — номер преобладающего коэффициента, $\upsilon_{\max}$ — максимальная частота, которую можно измерить (в зависимости от дискретизации), $k_{\max}$ — номер последнего коэффициента в буфере.

3.1. Анализ результатов (WAVE)

Как уже говорилось ранее, после того как преобразовали файлы из формата MIDI в формат WAVE, мы уже перестали иметь дело с символьными последовательностями. Теперь мы проанализируем те же мелодии, но с точки зрения спектрального анализа звуковых сигналов.

Сравнивая эталон и 1-ю мелодию, полученную из него вертикальным переносом на пять ступеней вверх, мы подвергаем преобразованию Адамара оба WAVE-файла. На рис. 1 изображена исходная мелодия после преобразования Адамара, стрелкой указан пик — преобладающий коэффициент (его значение — 5842).

Считаем по указанной выше формуле искомую частоту υ:

$$\upsilon = \frac{k \cdot \upsilon_{\max}}{k_{\max}},\ k = 5842,\ \upsilon_{\max} = 22050,\ k_{\max} = 262144.$$

Таким образом, искомая частота $\upsilon \approx 491.3$ Hz, что приблизительно соответствует частоте звука для ноты «си».

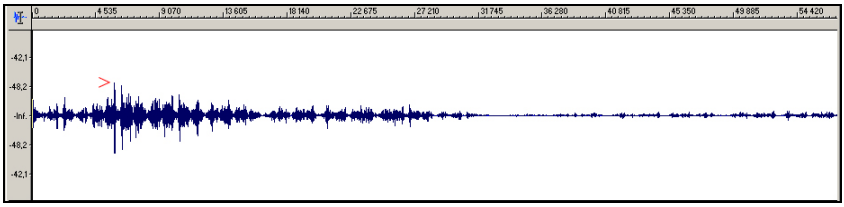


Рис. 1. Эталон после преобразования Адамара

Проделываем то же самое для 1-й трансформированной мелодии (рис. 2).

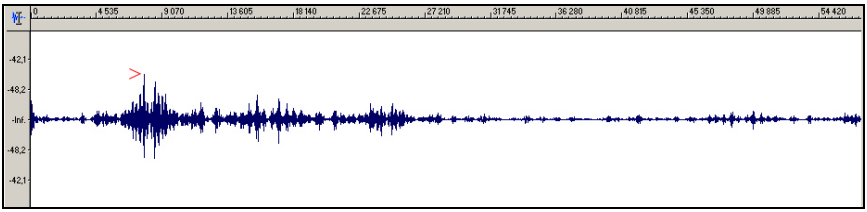


Рис. 2. Мелодия, трансформированная вертикальным переносом после преобразования Адамара

Теперь преобладающий коэффициент $k = 7829$, следовательно $\upsilon \approx 659$ Hz, что приблизительно соответствует частоте звука для ноты «ми».

Расстояние от ноты «си» вверх до ноты «ми» — как раз пять ступеней.

Посмотрим, как выглядит после преобразования Адамара 4-я трансформированная мелодия. Из всех четырех она максимально различается с эталоном.

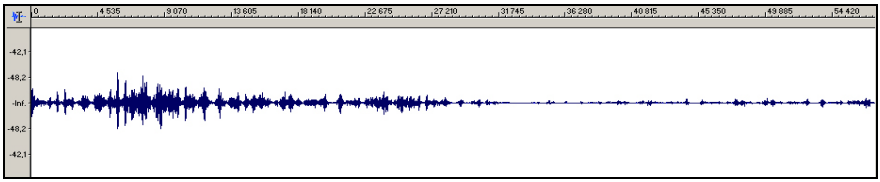


Рис. 3. 4-я трансформированная мелодия после преобразования Адамара

Здесь (на рис. 3) преобладающий коэффициент такой же, как и у эталона, поэтому общего переноса мелодии по вертикали вверх или вниз нет. Некоторые пики исчезли, некоторые добавились — это свидетельствует об

исчезновении и появлении некоторых частот, более конкретные выводы можно будет сделать только при доскональном анализе.

4. ЗАКЛЮЧЕНИЕ

Спектральный анализ звуковых сигналов с помощью преобразования Адамара позволяет легко выявлять изменения тембра или пока только трансформации 1-го типа, более сложные трансформации анализу подвергаются с очень большим трудом. И это в случае одноголосых мелодий. Для монофонических мелодий задача определения тембра решаема, но, когда мы будем иметь дело с полифоническими композициями с несколькими тембрами, такой анализ произвести будет практически невозможно. Также спектральный анализ бессилен в тех случаях, когда трансформироваться будет только ритмическая часть.

Для определения структуры мелодии лучше всего подходить к анализу с точки зрения символьных последовательностей. Для получения более достоверных данных необходимо модифицировать метрику таким образом, чтобы учитывать музыкальные правила, т.е. разным преобразованиям ставить в соответствие разные цены.

СПИСОК ЛИТЕРАТУРЫ

1. **Зарипов Р. Х.** Машинный поиск вариантов при моделировании творческого процесса. — М.: Наука, 1983.
2. **Духнич Е. И.** Об одном подходе к выполнению цифровых линейных преобразований. — Кибернетика. — 1981. — № 5.
3. **Рабинер Л., Шафер Р.** Цифровая обработка речевых сигналов. — М.: Радио и связь, 1981.
4. **Algolist** http://algolist.manual.ru
5. **ITMAN** http://itman.narod.ru

Шкурко Д. В.

О СЛОЖНОСТИ ПОСТРОЕНИЯ ОПТИМАЛЬНОГО ЛИНЕЙНОГО УЧАСТКА*

1. ВВЕДЕНИЕ

Оптимизация линейных участков занимает важное место в построении оптимизирующих трансляторов и, как правило, включается в набор оптимизирующих преобразований, выполняемых транслятором. От оптимизации линейных участков требуется, чтобы ее сложность была приемлемой, так как это всего лишь один из видов оптимизаций, производимых транслятором. К сожалению, этого можно достичь не всегда, и приходится выделять некоторый набор оптимизирующих приемов, который, возможно, не дает лучшего участка относительно данных преобразований и критерия оптимальности.

Как правило, при оптимизации программ интересуются двумя критериями: временем исполнения и объемом требуемой памяти. В данной работе внимание сконцентрировано на первом критерии.

Рассматриваемая модель линейных участков, введенная в [4], является расширением моделей из [1] и [6]. В модели из [1] рассматриваются только схемные преобразования, в [6] добавляется единственная интерпретированная операция пересылки. В модели [4] добавляются конструкции для описания структурных переменных, и в качестве интерпретации используются классы эквивалентности термов, чтобы описать повторные присваивания частям структурных переменных. Благодаря тому, что эта модель более выразительна, она потенциально лучше подходит для использования в реальных трансляторах. Среди существенных загрублений модели следует отметить отсутствие учета косвенной адресации ([2]).

Из соображений эффективности применения в модели из [4] были определены преобразования, имеющие явный оптимизирующий эффект. Аналогичный подход используется в [3], где, помимо полного набора преобразований, выделяются частные случаи общих преобразований, позволяющие эффективно искать возможности по их применению.

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

До конца не было ясно, какова сложность построения оптимального участка в рассматриваемой модели. В [5] была описана процедура оптимизации для сокращенного набора преобразований. В данной статье показана NP-полнота задачи оптимизации в общем случае. Такая сложность сохраняется и для более простой модели из [6].

2. МОДЕЛЬ ЛИНЕЙНЫХ УЧАСТКОВ И.В. ПОТТОСИНА

Фиксируется счетное множество Σ имен переменных и конечное множество Θ операций с заданнной арностью. В подмножестве Θ выделяется подмножество $\Theta' \subset \Theta$ легковычислимых функций. Пересылка θ_0 лежит в Θ'. Отдельно выделяется конечное множество функций выбора Λ.

Линейным участком, или *блоком,* называется тройка $\mathcal{B} = (P, V, U)$, где

- P — конечный список операторов $S_0; S_1; \ldots; S_n; S_{n+1}$ $(n \geq 0)$,
- $V \subset \Sigma$ — конечное множество *входных* переменных,
- $U \subset \Sigma$ — конечное множество *выходных* переменных.

Переменные, входящие в операторы и не лежащие в подмножествах U и V, называются *промежуточными.* Множество таких переменных обозначается W. Элементы множеств Σ, U, V, $U \cup V$ и W обозначаются далее, соответственно, буквами x, u, v, z и w с подходящими индексами.

Оператор S_0 — $start(v_1, \ldots, v_k)$, где $v_1, \ldots, v_k$ — все входные переменные. Оператор S_{n+1} — $finish(u_1, \ldots, u_k)$, где $u_1, \ldots, u_k$ — все выходные переменные. Остальные операторы имеют вид

$$A \leftarrow \theta B_1 \ldots B_r,$$

где $\theta \in \Theta$ и $B_1, \ldots, B_r$ — переменные или выборы аргумента (см. ниже), а A — переменная или выбор результата (см. ниже). Говорят, что B_i сопоставлены входам, а A — выходу.

Структурные переменные вводятся с помощью двух конструкций. Конструкция вида

$$g(z_1, \ldots, z_l)[z_{l+1} \ldots z_{l+k}], g \in \Lambda$$

называется *выбором аргумента,* если она сопоставлена входу; или *выбором результата,* если она сопоставлена выходу. g — функция выбора, $z_{l+1} \ldots z_{l+k}$ — *список аргументов* или *список результатов,* соответственно. Если переменная входит в список аргументов или результатов,

то считается, что в оператор она входит неявно, иначе — вхождение явное.

Если выходу оператора S_i сопоставлена переменная, то эта переменная является *обязательным* результатом, иначе, если сопоставлен выбор результата, множеством *необязательных* результатов является список результатов.

Аргументы оператора — это все переменные, сопоставленные входам, в объединении со всеми переменными из выборов аргументов и всеми переменными из выбора результата, неявляющимися результатами.

Отдельно определяются аргументы и результаты операторов S_0 и S_{n+1}. Обязательные результаты оператора *start* — все входные переменные, аргументов у *start* нет. Аргументы оператора $finish$ — все выходные переменные, результатов у $finish$ нет.

Аргументы оператора S_i обозначаются A_i, обязательные результаты — R'_i, а все результаты — $R_i \supset R'_i$.

Линейный участок задает вычисление некоторого терма для каждой выходной переменной. Линейные участки эквивалентны, если для каждой выходной переменной они вычисляют один и тот же терм. Этот терм задается с помощью интерпретации всех конструкций в множестве слов над алфавитом $V \cup \Theta \cup \Lambda$.

- В результате исполненения оператора *start* все входные переменные получают значения: $\mathrm{Val}_0(A) = A, A \in V$. Значения остальных переменных неопределеные.
- Функции выбора, входящей в S_i, сопоставляется слово

$$\mathrm{Val}_i\, g(z_1, \ldots, z_n) = g\, \mathrm{Val}_{i-1}\, z_1 \ldots \mathrm{Val}_{i-1}\, z_n.$$

- Входу оператора S_i сопоставляется $\mathrm{Val}_{i-1}(A)$, если вход — это переменная A, либо

$$\mathrm{Val}_i\, g(z_1, \ldots, z_l)[\mathrm{Val}_{i-1}\, z_{l+1} \ldots \mathrm{Val}_{i-1}\, z_{l+k}],$$

  если вход является выбором аргумента.
- Наконец, определяется, каким образом изменяются значения переменных после исполнения оператора $S_i = A \leftarrow \theta B_1 \ldots B_r$. Обозначим $\mathrm{Val}\, S_i = \theta\, \mathrm{Val}_{i-1}(B_1) \ldots \mathrm{Val}_{i-1}(B_r)$.
  - Если A — переменная, то $\mathrm{Val}_i(A) = \mathrm{Val}\, S_i$.
  - Если $A = g(z_1, \ldots, z_l)[z_{l+1} \ldots z_{k+l}]$ — выбор результата, то пусть $t_1 = \mathrm{Val}_i\, g(z_1, \ldots, z_l)$, $t_2 = \mathrm{Val}\, S_i$. Тогда, при условии, что подстрока вида jt_1t_2 не содержится в $\mathrm{Val}_{i-1}(z_{k+j})$,

$\mathrm{Val}_i\, z_{k+j} = \mathrm{Val}_{i-1}\, z_{k+j} \# j t_1 t_2$, в противном случае значение z_{k+j} не меняется.

Значения других переменных не меняются. Пересылка не меняет значений: $\theta_0 B = B$.

- Если значения каких-то аргументов оператора неопределены, то и значение оператора неопределено.
- Другим способом переменная получить значение не может.

Требуется, чтобы значения всех выходных переменных были определены на выходе.

Переменная A *активна* после некоторого оператора S_t, если

- $A \in R_i$,
- $A \notin R_{i+1}, \ldots, R_j$,
- $0 \le i \le t \le j \le n$,
- на нее ссылается оператор S_{j+1}, т.е. A является аргументом или необязательным результатом S_i.

Последовательность $S_{i+1}; \ldots; S_{j+1}$ с максимальным j называется *областью действия* переменной из R'_i, если выполнены условия

- $A \notin R'_{i+1}, \ldots, R'_j$,
- $0 \le i \le t \le j \le n$.

Область влияния $S_i (y \in R'_i)$ называется *замкнутой,* если ее последним оператором S_j является такой, что $y \in R'_j$, и операторы, входящие в область влияния S_i, не содержат неявных вхождений y. Участок без замкнутых областей влияния называется *открытым.*

Оператор S_k *непосредственно зависит* от $S_i (i < k)$, если

- S_k содержится в области влияния S_i по z и $z \in A_k$;
- S_k содержится в области влияния S_i по z и $z \in R_k$;
- S_i и S_k содержатся в области влияния $S_j (j < i)$ по z и $z \in A_i \cap R_k$.

В 1-м и во 2-м случаях, при условии, что $z \notin R'_k$, зависимость является информационной, а в последнем обусловлена тем, что неявное использование требует конкретной переменной и эта переменная не может быть заменена на другую с тем же значением. Транзитивное замыкание отношения непосредственной зависимости дает отношение зависимости.

Часть неинформационных зависимостей может быть удалена с помощью следующего преобразования.

Контекст. Оператор S_i в программе таков, что область влияния по его выходной переменной x замкнута.

Преобразование 1. *Выбирается новая промежуточная переменная w, она сопоставляется выходу S_i, и все вхождения y в области*

влияния S_i заменяются на w.

Преобразование 2 (удаление бесполезных присваиваний). *Если среди результатов некоторого оператора нет активных переменных, тогда его можно удалить. Или если оператор является пересылкой и не меняет значения выходной переменной, то его тоже можно удалить.*

Преобразование 3 (исключение избыточных вычислений). *Пусть $S_i = A \leftarrow \theta B_1 \dots B_r$, $S_j = A' \leftarrow \theta B'_1 \dots B'_r$, $i < j$ $\mathrm{Val}\, S_i = \mathrm{Val}\, S_j$ и $\theta \in \Theta - \Theta'$. Тогда вводится новая промежуточная переменная w, и S_i заменяется на пару операторов $w \leftarrow \theta B_1 \dots B_r, A \leftarrow \theta_0 w$, а S_j заменяется на пересылку $A' \leftarrow \theta_0 w$.*

Остальные преобразования касаются только удаления избыточных пересылок.

Контекст. S_i — пересылка $w \leftarrow \theta_0 x$. S_j — оператор с наименьшим номером $j \geq i$, такой что $x \in R_j$. Если выделить в P подпоследовательность $S = S_{k_1}, \dots, S_{k_l} (j < k_1 < \dots < k_l)$ такую, что S_{k_r} — либо оператор, в котором w имеет явное вхождение, либо от которого зависит такой оператор, причем в S входят все такие операторы, то все операторы из S не зависят от S_j.

Преобразование 4. *Подпоследовательность S перемещается перед S_j. Удаляется S_i, и все вхождения w в S заменяются на x.*

Контекст. S_i — пересылка $x \leftarrow \theta_0 w$. S_j — единственный оператор, который присваивает w $(i > j)$. Если выделить в P подпоследовательность $S = S_{k_1}, \dots (j < k_1 < \dots < i)$, состоящую из всех операторов между S_i и S_j таких, что либо $x \in A_{k_r} \cup R_{k_r}$, либо от него зависит такой оператор, то все операторы из S не зависят от S_j. Далее выделяется подпоследовательность $S' = S_{q_1}, \dots (q < q_1 < \dots)$ и q — минимальный номер такой, что $x \in R_q$, S_{q_i} зависит от w, либо от него зависит такой оператор, и все операторы S' не зависят от S_q.

Преобразование 5. *Последовательности S и S' перемещаются соответственно перед S_j и перед S_q. Из полученной программы удаляется пересылка S_i. В S_j выходная переменная заменяется на x, и все вхождения w в области влияния S_j заменяются на x.*

3. ОПТИМИЗАЦИЯ

Пусть входу всех пересылок сопоставлены переменные, т.е. выборы аргументов всегда перевычисляются.

С каждым линейным участком связывается граф зависимостей, являющийся расширением информационного графа. Дуги этого графа, определяющие отношение непосредственной зависимости, помечаются именем переменной, а также характером зависимости: информационная или нет.

Лемма 1. *Если задан бесконтурный граф, который размечен таким образом, то можно построить его укладку с сохранением имен переменных, по которым осуществляется зависимость.*

◀ надо только заметить, что обычная укладка удовлетворяет данному свойству. ▶

В [5] введено понятие тупиковости применения преобразования T_1 по отношению к применению T_2.

Если к участку применимы $T_1(S_j)$ и $T_2(S_i)$, а после того как $T_1(S_j)$ проделано, $T_2(S_i)$ уже не применимо, будем говорить, что T_1 *тупиково* к T_2. Далее рассматривается влияние применения всех преобразований на построенный граф. Добавленные в результате применения преобразования дуги назовем *существенными,* если они создают путь между вершинами, которые до преобразования не зависели одна от другой. Существенные дуги могут определять только неинформационную зависимость.

Утверждение 1. *Преобразование T1 удаляет неинформационные зависимости. Поэтому оно не тупиково к T2, T3, T4 и T5. Значит, можно считать, что участок открытый, и после каждого T2, T3, T4 или T5 участок преобразуется к открытому.*

Отсюда, в частности, следует, что единственность оператора, присваивающего w в преобразовании T5, несущественна. Этого всегда можно добиться с помощью T1 (у промежуточных переменных нет неявных вхождений). Можно также считать, что промежуточным переменным присваивается значение только один раз.

Утверждение 2. *Преобразование T2 удаляет вершины и дуги, которые инцидентны этим вершинам. Значит, далее можно считать, что сначала удаляются все бесполезные вычисления.*

◄ Первая часть утверждения очевидна. Для доказательства второй части следует заметить, что оптимальность участка определяется числом вершин графа, а применимость всех преобразований — дугами. ►

Утверждение 3. *Преобразование T3, примененное к S_i и S_j, не удаляет операторы и не добавляет существенные дуги, если аргументы, не входящие в выборы результатов, вычисляются одними и теми же операторами.*

Утверждение 4. *После T4 могут возникнуть неинформационные зависимости. Оператор, который зависел от пересылки (т.е. присваивал значение u), будет зависеть от всех операторов, использующих w. Других существенных дуг не будет.*

◄ Зависимость операторов определяется только аргументами и результатами. Значит, утверждение следует из разбора случаев непосредственной зависимости операторов. ►

Утверждение 5. *Пусть в контексте преобразования T5 пересылка неинформационно зависит от использования или присваивания переменной u, тогда от него будет зависеть оператор S_j. От операторов, использовавших w, будут теперь неинформационно зависеть операторы, которые до преобразования неинформационно зависели от пересылки.*

◄ Полностью аналогично предыдущему. ►

Утверждение 6. *Преобразования T4 и T5 ничего не добавляют, если обе переменные промежуточные, и такую пересылку можно удалить в любой момент с помощью T4 или T5. Результат от применения T4 такой же, с точностью до переименования, как и от применения T5.*

◄ Здесь используется тот факт, что с помощью T1 участок приводится к эквивалентному, в котором присваивание промежуточной переменной делается только один раз. ►

Далее рассматриваются максимальные по включению последовательности преобразований.

В процессе применения преобразований может происходить добавление операторов (только в результате преобразования T3), а также перестановка операторов. Не уменьшая общности, можно предполагать,

что операторы пронумерованы таким образом, что оператор, информационно зависящий от другого, имеет больший номер. Это возможно, так как ни одно преобразование не меняет информационных зависимостей.

Лемма 2. *Пусть дана последовательность*

$$T_1, T_2, \ldots, T_n, \tag{1}$$

которая преобразует $\mathcal{B} = (P, V, U)$ *в* $\mathcal{B}' = (P', V, U)$, $\Theta' = \{\theta_0\}$ *и* T_i — *преобразование Т3, примененное к паре операторов с наименьшей парой, относительно следующего упорядочения (лексикографического):*

$$(i, j) < (i', j'), \text{ если } i < i' \text{ или } i = i', j < j'.$$

Тогда если $\mathcal{B} \xrightarrow{T_i} \mathcal{B}''$, *то среди всевозможных участков, которые могут быть получены из* $\mathcal{B}''$, *есть* $\mathcal{B}'$.

◀ Пусть T_i применяется к S_i и S_j $(i < j)$, тогда можно утверждать следующее: аргументы левой части S_j получаются из аргументов S_j с помощью пересылок, и их число совпадает, так как выборы аргументов всегда перевычисляются. В противном случае можно было бы найти пару с меньшим номером. Значит, как и в утверждении Т3, все пути, по которым осуществляется зависимость в S_j, могут только укоротиться:

$$T5T_i = T_iT5$$
$$T4T_i = T_iT4$$
$$T3T_i = T_iT3.$$

Из этих равенств следует утверждение о выносе в начало преобразования T_i. ▶

Замечание 1. *Если считать, что выборы аргументов не перевычисляются или есть легко вычислимая функция, кроме пересылки, то предыдущая лемма уже не будет верна, так как Т3 может добавлять существенные дуги.*

Теперь можно считать, что вначале применяются операторы Т3 с убывающими номерами пар операторов, к которым они применяются. После того как применили все Т3, получится следующая ситуация: левая часть всех операторов, кроме пересылок, вычисляет уникальное значение, которое потом с помощью нескольких пересылок передается в те операторы $\mathcal{B}$, которые их используют.

Теорема 3. *Если* $\Theta' = \emptyset$ *или* $\Theta' = \{\theta_0\}$, *оптимизацию можно сделать так:*

- *сначала применяются всевозможные T2, как указано выше;*
- *потом все T3;*
- *наконец, все T4 и T5.*

Удаление тривиальных пересылок, по утверждению 6, можно делать на любом шаге.

◀ Следует из утверждений 2 и 6 и леммы 2. ▶

Теорема 4. *Задачу о размыкании всех контуров ориентированного графа* G *можно за полиномиальное время свести к задаче об оптимизации участка, в котором имеются неявные вхождения переменных, и есть, по крайней мере, одна операция, отличная от пересылки.*

◀ Каждой вершине графа G будет соответствовать вычисление разных выходных переменных из разных входных, причем эти вычисления вначале не связаны. Каждой дуге графа G будет соответствовать пересылка, к которой применимо T4, эта пересылка при удалении создает только одну существенную дугу. Эта дуга соответствует дуге данного графа. В некоторых случаях потребуется, чтобы число аргументов оператора было больше 1, это достигается использованием выборов аргументов.

Построение проводится постепенно, добавлением одной дуги за раз. Добавление дуги (a, b) происходит следующим образом. Если уже построены дуги, соответствующие дугам (x, a), то новая дуга начинается в операторе, достижимом из всех операторов, являющихся концами существенных дуг, соответствующих построенным дугам (x, a). Аналогично, операторы начала дуг (b, y), должны быть достижимы из конца новой существенной дуги.

Несложно заметить, что оптимизация свелась к удалению пересылок, а удаление пересылок эквивалентно применению максимального числа преобразований T4, что, в свою очередь, эквивалентно удалению минимального количества ребер графа G для размыкания всех контуров. Контекст T3 не выполняется никогда, так как выходные переменные зависят от разных входных переменных. Для каждой пересылки $w \leftarrow \theta_0 v$ требуется одна входная переменная v, одно использование промежуточной переменной w, одно неявное использование v, после одного присваивания v. Отсюда следует утверждение. ▶

Теорема 5. *В общем случае (не налагаются никакие условия на Θ' и на использование выбора результата) оптимизация делается за полиномиальное время недетерминированным алгоритмом.*

◀ Длина цепочки преобразований, которые могут быть применены к участку, ограничена числом $4n$, где n — длина программы. Два раза к одному оператору можно применить различные преобразования, только если первое было Т3. Добавляются операторы только с помощью Т3, и число новых операторов не превышает n. ▶

Следствие 1. *Задача оптимизации NP-полная.*

4. МОДЕЛЬ С. МАТВИНА

Лемма 6. *У открытых участков графы изоморфны с сохранением разметки тогда и только тогда, когда любой из них можно преобразовать в другой при помощи перестановки соседних независимых операторов и переименования промежуточных переменных.*

◀ Достаточность очевидна. Требуется доказать необходимость. Для каждой перестановки $i_1 \dots i_n$ чисел $1 \dots n$ вводится четность — число пар чисел от $(i_k, i_j), 1 \le k < j \le n$ и при этом $i_k > i_j$. Теперь операторы первого блока нумеруются, и рассматривается перестановка, соответствующая второму блоку. Если четность равна 0, то после изменения имен промежуточных переменных получаются одинаковые участки. Если четность не равна 0, то во втором участке есть стоящие рядом операторы с номерами i и j, причем первым стоит оператор с большим номером. Эти операторы независимы вследствие изоморфизма графов, значит, их можно переставить и уменьшить четность на единицу. Такие пары операторов ищутся следующим образом: выделяются группы операторов с возрастающими номерами операторов и берутся пограничные операторы. Они всегда есть, если число групп больше одной, что эквивалентно тому, что четность отлична от нуля. ▶

Следствие 2. *Если допустить перестановку соседних независимых операторов, то в схеме оптимизации в теореме 3 ничто не изменится.*

Если в участке нет выборов результатов и аргументов, преобразования можно применять в любую сторону и $\Theta' = \{\theta_0\}$, то получается модель, введенная Матвиным в [6]. Далее рассматривается эта модель.

Предполагается известным тот факт, что любой участок, эквивалентный данному в смысле определения, можно преобразовать к данному с помощью преобразований Т2 и Т3. Доказательство приводится, например, в [3].

Утверждение 7. *Можно считать, что каждая переменная появляется в левой части присваивания не более одного раза.*
◀ Это достигается переименованием переменных, которое является комбинацией преобразований Т2 и Т3. ▶

Теперь по участку строится граф G. Вначале рассматривается информационный граф G' выражений, в нем сразу же удаляются вершины, не используемые для вычисления значений выходных переменных. Далее каждой вершине этого графа приписываются имена переменных, в которых должно оказаться соответствующее выражение (входной вершине приписывается также соответствующая входная переменная). Требуется дополнить граф G' неинформационными зависимостями, аналогичными зависимостям из определения 2. Эти зависимости вследствие утверждения 7 могут появиться только для переменных из $U \cap V$. Переменные из $U \cap V$ называются далее *транзитными.* Точнее, оператор, которому приписана транзитная переменная, зависит неинформационнно от каждого оператора, использующего входное значение этой переменной, всем таким дугам соответствует данная переменная. Петли не добавляются. В итоге получается граф G.

Каждой вершине соответствует одно вычисление (либо оператор **старт**, если вершине отвечает входная переменная) и $n-1$ пересылка, где n — число переменных, приписанных вершине.

Так как граф зависимостей по аргументам бесконтурный, то в любом контуре графа G есть дуга неинформационной зависимости.

Теперь сформулируем критерий, когда можно построить по графу G участок. Пусть дана укладка графа G. Дуга неинформационной зависимости называется *фиктивной,* если оператор ее начала не использует (у входной переменной есть копия) или оператор ее конца не присваивает соответстсвующей переменной. Пусть задан граф G, удовлетворяющий условиям:

- каждой дуге приписано r или a (r соответствует неинформационной зависимости, a — информационной);
- каждая вершина помечена либо как операция $\theta \in \Theta$, либо как входная переменная;

- зафиксирован порядок a-дуг, входящих в каждую вершину;
- граф G', получающийся удалением всех r-дуг из G, бесконтурный;
- каждой вершине, помеченной операцией $\theta \in \Theta$, приписан список переменных, причем каждая переменная может появляться в нем не более одного раза и только, если она промежуточная или выходная;
- r-дуги определены так же, как и ранее, и им приписаны имена транзитных переменных.

Утверждение 8. *По графу G, описанному выше, можно построить участок, если, и только если, в каждой вершине можно выбрать приписанную переменную так, чтобы в любом контуре имелась фиктивная r-дуга.*

◀ Достаточность. Очевидный факт, так как иначе r-дуги накладывают ограничение на порядок расположения операторов.

Необходимость. Удалив фиктивные дуги, получаем бесконтурный граф, по которому строится участок. Выбранной переменной присваивается значение каждой вершины, которое пересылается в остальные переменные в конце. Для входных вершин это правило изменяется так: если входной вершине приписана промежуточная переменная, то в ней вначале сохраняется значение входной переменной для последующего использования. Корректность проверяется тривиально. ▶

Теперь можно оптимизировать участок. Сначала рассматриваются все операторы, которые нельзя исключить. Для этого в каждой вершине оставляются только выходные переменные, если таких нет, то единственная промежуточная переменная, приписанная вершине, остается (получается граф G_{st}). Если при такой разметке можно выбрать переменые, как в утреждении 8, то полученный участок будет оптимальным. Если нет, то требуется приписать минимальное количество новых промежуточных переменных некоторым вершинам, чтобы это стало возможным.

Теорема 7. *Построенный таким образом участок будет оптимальным относительно обоих критериев, указанных в [6].*

◀ Сравним граф оптимального относительно любого критерия участка с построенным выше. Они будут содержать одинаковое число вершин. Значит, осталось минимизировать только число пересылок, но

в построенном участке оно минимально. Следовательно, построенный участок оптимальный относительно обоих критериев. ►

Утверждение 9.

1) Если в некоторой вершине a графа G_{st} есть нетранзитная переменная u, то можно считать, что значение вершины присваивается u, поэтому r-дуги, входящие в нее, всегда рассматриваются как фиктивные и удаляются.

2) Если через некоторую r-дугу D не проходит ни один контур, то все r-дуги, входящие в ее конец a, можно удалить, так как выходной переменной оператора вершины a будет переменная, соответствующая дуге D.

◄ Просто требуется заметить, что если присваивать, как указано в условиях, то не добавится контуров без фиктивных r-дуг. ►

Лемма 8. *Если в графе G оптимизированного участка некоторой входной вершине v приписана промежуточная переменная, то, удалив v и приписав новую переменную вершине θ, соответствующей выходному значению v, получится граф некоторого другого оптимального участка.*

◄ Приписыванием промежуточной переменной входной вершине v получается, что можно считать фиктивными только r-дуги, соответствующие v. Если же добавить новую переменную в вершину θ, то фиктивными станут еще и r-дуги, соответствующие другим переменным, приписанным θ. ►

Далее рассматривается вспомогательный мультиграф H. Его вершинами будут все невходные вершины графа G_{st}, не удовлетворяющие условиям утверждения 9. Две вершины t и z графа H соединяются дугой (t, z), если существует путь из t в z в графе G_{st}, состоящий из a-дуг и r-дуг, соответствующих некоторой фиксированной переменной u, приписанной z. Эта дуга помечается именем u. В мультиграфе H могут быть петли.

В соответствии с утверждением 8 требуется введение новых переменных, для того чтобы в графе G_{st} было достаточно фиктивных дуг для размыкания всех контуров. Введение новых переменных моделируется упрощениями мультиграфа H.

1) Выбирается и удаляется некоторая вершина и все ребра, инцидентные ей, в H (это соответствует приписыванию соответствующей вершине графа G_{st} новой промежуточной переменной).
2) Если появились вершины t без входящих ребер, помеченных некоторой переменной, приписанной t, то t удаляется вместе со всеми инцидентными ребрами. Это проделывается пока возможно.

Теорема 9. *Задача оптимизации сводится к минимизации числа упрощений для получения пустого графа.*

◀ По утверждению 9 и лемме 8 новые промежуточные переменные добавляются только в вершины графа H. Если мультиграф H станет пустым, то по графу G_{st} можно построить участок. С каждой вершиной H, а с ней и вершиной графа G_{st} связывается либо новая переменная, либо переменная, для которой не было входящих дуг во время выполнения какой-нибудь очередной операции (см. утверждение 9).

Если мультиграф H после некоторого шага не пуст, то по графу G_{st} невозможно построить участок. Доказательство проводится от противного. Рассматриваются все операторы, которые соответствуют вершинам H. Для каждого из них есть предшествующий, так как есть входные дуги мультиграфа H, соотвествующие всем переменным, приписанным вершине графа G_{st}. В этом состоит противоречие.

Число операций над мультиграфом H равно количеству пересылок, добавленных для того, чтобы не осталось контуров графа G_{st} с фиктивными дугами. ▶

Следствие 3. *Если в графе G_{st} каждой вершине приписана только одна переменная, то задача оптимизации сводится к поиску минимального числа вершин графа H (в нем не будет кратных ребер), размыкающих все контуры H. Задача оптимизации NP-полная.*

Утверждение 10. *Построение мультиграфа H можно проделать за время $O(n^3)$, где n — длина программы.*

◀ Построение графа G_{st} делается за время $O(n^2)$, а по графу G_{st} мультиграф H строится за время $O(n^3)$. ▶

Очевидно, что можно построить недетерминированный алгоритм оптимизации. Отсюда следует NP-полнота.

5. ЗАКЛЮЧЕНИЕ

В работе показано, что оптимизация линейных участков в некоторых случаях не может быть выполнена за полиномиальное время, и поэтому требуется формулировка условий, при выполнении которых возможна полная оптимизация. Показано, что для участка С. Матвина сложность определяется числом транзитных переменных, а в участке И. В. Поттосина — числом пересылок в исходном участке.

Приближенный алгоритм задачи оптимизации можно строить по приближенному алгоритму решения соответствующей задачи теории графов, так как построенные алгоритмы сведения не приводят к увеличению абсолютной погрешности приближенного решения задачи оптимизации по сравнению с погрешностью решения задачи теории графов.

СПИСОК ЛИТЕРАТУРЫ

1. **Ахо А., Ульман Дж.** Теория синтаксического анализа, перевода и компиляции: Пер. с англ. — М.: Мир, 1978. — Т. 2.
2. **Касьянов, В. Н.** Оптимизирующие преобразования программ. — М.: Наука, 1988. — 336 с.
3. **Касьянов, В. Н.** Эквивалентные преобразования линейных участков программ // Трансляция и преобразования программ. — Новосибирск, 1984. — С. 56–61.
4. **Поттосин, И. В.** Направленные преобразования линейного участка // Языки и системы программирования. — Новосибирск, 1981. — 17 с.
5. **Поттосин, И. В.** Об алгоритме оптимизации линейных участков // Трансляция и преобразования программ. — Новосибирск, 1984. — 14 с.
6. **Matwin S.** On the completeness of a set of transformations optimizing linear programs // Inf. proces. letters. — 1977. — Vol. 6. — P. 165–167

СОДЕРЖАНИЕ

CONTENTS

УДК 519.68 + 519.765

К вопросу об анализе текстов на естественном языке / Батура Т. В., Еркаева О. Н., Мурзин Ф. А. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 7–58.

Дается обзор различных методов и конструкций математической и классической лингвистики, применяемых для анализа текстов на естественном языке. — Библиогр.: 5 назв.

To the problem of analysis of texts in a natural language / Batura T. V., Erkayeva O. N., Murzin F. A. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 7–58.

The article presents an overview of various methods and constructions of mathematical and classical linguistics applied to analysis of texts in a natural language. — Refs: 5 titles.

УДК 519.68 + 519.767.02

Логические методы представления смысла текста на естественном языке / Батура Т. В., Мурзин Ф. А. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 59–111.

Целью данной работы является разработка разнообразных алгоритмов сопоставления предикатов и формул исчисления предикатов первого порядка текстам на естественном языке. Результаты работы могут быть применены в автоматизированных системах обработки текстов на естественном языке и для построения теории смысла текстов, что является предметом исследований, прежде всего, в лингвистике, а также в области математической логики. — Библиогр.: 7 назв.

Logical methods of the sense representation for a text in a natural language / Batura T. V., Murzin F. A. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 59–111.

The purpose of the work is the development of various algorithms of mapping the predicates and formulas of the first-order predicate calculus to the texts in a natural language. Its results can be used in the computer-aided systems for processing the texts in a natural language, as well as in constructing the theory of the text sense, which is a subject of research, first of all, in linguistics and also in mathematical logic. — Refs: 7 titles.

УДК 519.68 + 519.6

О некоторых задачах, связанных с автоматизацией процесса ядерного каротажа нефтяных скважин / Винокуров А. А., Ильин И. В., Лобив И. В., Мурзин Ф. А., Половинко О. Н., Семич Д. Ф. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 112–123.

Описываются математические постановки нескольких задач, возникающих при автоматизации процесса ядерного каротажа, и указаны методы их решения. — Библиогр.: 6 назв.

Some problems of automation of nuclear oil-wells carotage / Vinokurov A. A., Ilyin I. V., Lobiv I. V., Murzin F. A., Polovinko O. N., Semich D. F. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 112–123.

The paper presents the mathematical statements of several problems arising in automation of the process of nuclear carotage and the methods of their solution. — Refs: 6 titles.

УДК 519.68 + 681.3.06

Виртуальный музей истории информатики в Сибири: модель предметной области и модель пользователя / Волянская Т. А. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 124–146.

В статье представлен проект создания виртуального музея истории информатики в Сибири, разрабатываемого в виде информационно-поисковой, справочной адаптивной гипермедиа-системы, доступной в Интернете. Кратко описаны методы и технологии адаптивной гипермедиа, архитектура адаптивных гипермедиа систем и ее основные компоненты (модель предметной области, модель пользователя и модель адаптации), рассматриваются вопросы построения онтологии и модели предметной области, а также моделирования пользователя. — Библиогр.: 17 назв.

Virtual museum of informatics history in Siberia: the models of a domain and a user / Volyanskaya T. A. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 124–146.

The paper presents a project of a virtual museum of informatics history in Siberia developed as on-line information retrieval adaptive hypermedia system. It briefly describes the methods and techniques of adaptive hypermedia, the architecture of adaptive hypermedia systems and its components (the models of a domain, a user and adaptation); the problems of constructing the domain ontology, domain model and user model are also considered. — Refs: 17 titles.

УДК 519.68 + 681.3.06

Визуализация генетической информации / Дунаев А. А., Кель А. Э., Лобив И. В., Мурзин Ф. А., Половинко О. Н., Черемушкин Е. С. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 147–156.

Описаны несколько алгоритмов визуализации генетической информации и кратко сообщается о реализованных программах и тестах. — Библиогр.: 8 назв.

Visualization of genetic information / Dunaev A. A., Kel A. E., Lobiv I. V., Murzin F. A., Polovinko O. N., Cheryomushkin E. S. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 147–156.

The paper describes some algorithms of the genetic information visualization and presents information on implementation of some programs and tests. — Refs: 8 titles.

УДК 519.68 + 681.3.06

Применение вейвлет-преобразований в кодировании изображений / Иванов М. А. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 157–175.

Изложены основы теории вейвлет-преобразований. Рассмотрены основные методы применения вейвлет-преобразований к кодированию изображений: вейвлет-разложение, построение пакетов вейвлетов, построение нуль-деревьев, кодирование нуль-деревьев. — Библиогр.: 5 назв.

Application of wavelet-transforms to image coding / Ivanov M. A. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 157–175.

The paper gives the basis of the wavelet-transform theory and the main applications of wavelet-transforms to image coding, such as wavelet-decomposition, packets of wavelets, zero-tree construction, zero-tree coding. — Refs: 5 titles.

УДК 519.68 + 681.3.06

Упорядоченные диаграммы бинарных решений / Касьянов В. Н., Мирзуитова И. Л. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 176–219.

Упорядоченные бинарные диаграммы решений (OBDD) представляют булевы функции в виде ориентированных ациклических графов. Они образуют каноническое представление, с помощью которого проверка таких функциональных характеристик, как выполнимость и эквивалентность, может быть произведена простым образом. В данной статье рассматриваются OBDD-диаграммы, описываются алгоритмы их построения и обработки, приводится обзор задач, которые были решены с помощью символьного анализа, основанного на OBDD-представлениях. — Библиогр.: 37 назв.

Ordered binary-decision diagrams / Kasyanov V. N., Mirzuitova I. L. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 176–219.

Ordered Binary-Decision Diagrams (OBDDs) represent Boolean functions as directed acyclic graphs. They form a canonical representation, which makes testing of functional properties, such as satisfiability and equivalence, straightforward. In the paper, the OBDD representations are considered, the algorithms used to construct and manipulate them are described, a survey of applications solved by OBDD-based symbolic analysis is given. — Refs: 37 titles.

УДК 519.68 + 681.3.06

Алгоритмы поиска доминаторов в управляющем графе / Лакийчук О. А. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 220–230.

В статье рассмотрены некоторые существующие алгоритмы поиска доминаторов в управляющем графе. Представлен новый линейный алгоритм поиска доминаторов. — Библиогр.: 7 назв.

Algorithms for finding dominators in a flow graph / Lakychuk O. A. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 220–230.

Some available algorithms for finding dominators in a flow graph are considered and a new linear-time algorithm for finding dominators is presented. — Refs: 7 titles.

УДК 519.68 + 681.3.06

Программные средства для автоматического формирования тематической коллекции по преобразованиям программ для коллективного использования / Малинина Ю. В. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 231–245.

В последние годы развитие всемирной компьютерной сети привело к тому, что Интернет становится основным каналом опосредованной передачи всех видов информации, в том числе научной и технической. В связи с этим все более актуальными становятся проблемы исследования и составления коллекций научных документов по определенной тематике и полных тематических библиографий. Результаты исследований позволяют представить систему научной коммуникации как информационное пространство, где возникают и формируются научные тематики, и организовать эффективный поиск информации по определенной тематике.

В данной работе описывается один из подходов, предназначенный для автоматического формирования тематической коллекции по преобразованиям программ. — Библиогр.: 11 назв.

Program means of automatic generation of a thematic collection on program transformations for shared use / Malinina Ju. V. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 231–245.

Rapid growth of the world-wide web resulted in the fact that Internet is now the main channel of information exchange of all types, including scientific and technical. So, the problem of accumulation of the scientific documents in collections is becoming actual. The results of investigations in this field allow us to represent a system of scientific communication as an information space and to organize efficient information retrieval through the collections of documents on some specific subject.

This article describes an approach intended for automatic accumulation of documents in collections on program transformations. — Refs.: 11 titles.

УДК 519.68 + 681.3.06

Решение задачи нахождения оптимального положения тела в пространстве по данным, поступающим с одномерных камер, для 3D оптической системы анализа движения обьектов / Мехонцев Д. Ю., Лобив И. В., Мурзин Ф. А. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 246–253.

Задача восстановления положения тела по данным поступающим с видеокамер хорошо известна в кинематографе и компьютерных играх. В данной статье предложен алгоритм восстановления положения тела по данным, поступающим с одномерных камер. Работа выполнена для системы реального времени. — Библиогр.: 7 назв.

Solving “restore the body position by data from 1D-cameras” problem for a 3D optical system of object movement analysis / Mekhontsev D. Yu., Romanuta M. A., Seleznev K. S., Lobiv I. V., Murzin F. A. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 246–253.

Restoring a body position by data from video cameras is a well-known task in computer games and cinema. In the paper, an algorithm of restoring a body position by one-dimensional cameras is suggested. The work is performed for a real-time system. — Refs: 7 titles.

УДК 519.68 + 681.3.06

Светосильные растровые структуры для режима автоколлимации / Мурзин Ф. А., Мурзина Т. С., Хаяров Е. М., Шлишевский В. Б. // Новые ин-

формационные технологии в науке и образовании. — Новосибирск, 2003. — С. 254–255.

Сообщается, что в результате проведенных теоретических исследований и численного моделирования предложен алгоритм построения высокосветосильных растровых структур для режима автоколлимации с разностными автокорреляционными функциями с очень малыми побочными максимумами. Результаты работы могут быть использованы при конструировании растровых спектрометров. — Библиогр.: 4 назв.

High luminosity raster structures for an autocollimation regime / Murzin F. A., Murzina T. S., Khayarov E. M., Shlishevsky V. B. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 254–255.

The paper presents an algorithm of constructing high luminosity raster structures for an autocollimation regime with difference autocorrelation functions having very small accessory maximums. The results can be used in constructing raster spectrometers. — Refs: 4 titles.

УДК 519.68 + 681.3.06

Распознавание текстур по пространственным закономерностям / Мурзин Ф. А., Половинко О. Н., Лобив И. В. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 256–268.

В статье описаны результаты исследования задачи распознавания текстуры по некоторому её участку.

Рассмотрены три метода, основанные на анализе плотностей перепадов яркости, автокорреляционных функций и различных статистических характеристик серий. Также проводится анализ закономерностей полученных характеристик текстурного изображения. — Библиогр.: 4 назв.

Texture recognition using spatial characterisitics / Murzin F. A., Polovinko O. N., Lobiv I. V. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 256–268.

This paper deals with the pattern recognition problem for textures. Three methods are considered: the distribution analysis for brightness difference, the method that uses the autocorrelation function, and the method based on statistic characteristics of series. The texture analysis of characteristics' properties is described. — Refs: 4 titles.

УДК 519.68 + 681.3.06

Исследование алгоритмов варьирования применительно к MIDI-файлам / Мухин Д. Ю. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 269–279.

Рассматривается задача нахождения степени сходства между трансформированными мелодиями и мелодией-эталоном и анализа степени изменения в зависимости от произведенных трансформаций мелодии. — Библиогр.: 5 назв.

Investigation of variation algorithms applied to MIDI-files / Mukhin D. Yu. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 269–279.

The paper considers the problem of finding a degree of similarity between the transformed melody and the melody-sample and analysis of a degree of modification depending on the melody transformations. — Refs: 5 titles.

УДК 519.68 + 681.3.06

О сложности построения оптимального линейного участка / Шкурко Д. В. // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 280–294.

Рассматриваются проблемы оптимизации линейных участков в построении оптимизирующих трансляторов. Обсуждаются критерии оптимизации программ. Показана NP-полнота задачи оптимизации в общем случае. — Библиогр.: 6 назв.

On complexity of construction of an optimal linear interval / Shkurko D. V. // New informational technologies in science and education. — Novosibirsk, 2003. — P. 280–294.

The paper considers the problem of linear interval optimization in the optimizing compiler construction. It discusses the criteria of program optimization and shows that the optimization problem is NP- complete in a general case. — Refs: 6 titles.

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В НАУКЕ И ОБРАЗОВАНИИ

Под редакцией
проф. Виктора Николаевича Касьянова

Рукопись поступила в редакцию 01. 07. 2003

Ответственный за выпуск Г. П. Несговорова
Редактор З. В. Скок

Подписано в печать 27. 12. 2003
Формат бумаги 60 × 84 1/16 Объем 17,3 уч.-изд.л., 18,9 п.л.
Тираж 75 экз.

НФ ООО ИПО "Эмари" РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6