ISSN 0136-3549
0320-3409

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED
568

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

TPI '84

ОБРАБОТКА ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ, ВОПРОСЫ ПРОГРАММИРОВАНИЯ

568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

ТПИ '84

УДК 518.5:519.2
681.03

ОБРАБОТКА ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ, ВОПРОСЫ ПРОГРАММИРОВАНИЯ

Труды экономического факультета LI

Таллин 1984

Таллинский политехнический институт
Труды ТПИ № 568
ОБРАБОТКА ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ,
ВОПРОСЫ ПРОГРАММИРОВАНИЯ
Труды экономического факультета L1
На русском языке
Ответственный редактор И. Амитан.
Технический редактор В. Ранник.
Сборник утвержден коллегией Трудов ТПИ 13.01.84.
Подписано к печати 28.06.84.
МВ-00599.
Формат 60x90/16.
Печ. л. 7,25 + 0,25.
Уч.-изд. л. 6,5.
Тираж 300.
Зак. № 432.
Цена 1 руб.
Ротапринт ТПИ, Таллин, ул. Коскла, 2/9.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 518.5:519.2

Л.К. Выханду, П.Л. Выханду

СИНТЕЗ МЕТОДА АДРЕСНЫХ КНИГ И РАСШИРЯЮЩЕГОСЯ ХЭШИРОВАНИЯ

I. Введение

В данной статье предлагается метод быстрого поиска объектов в больших наборах данных. Для этого используются техники создания адресных книг и расширяющегося хэширования.

Начальные данные представлены в виде т.н. битовых изображений, на которых создают адресные книги [1]. При использовании последних наборы данных можно разбить на естественные группы. Это дает возможность в ходе поиска определять группы записей, не подлежащих более подробному изучению. Описание этой техники изложено в статье [1], поэтому в данной статье приводится только краткий обзор ее.

Особое внимание обращается на расширяющееся хэширование. При обыкновенном хэшировании длина хэштаблицы постоянна и зависит от функции расстановки и числа записей. Слишком большая хэштаблица приводит к значительной затрате памяти, а малая таблица требует дорогостоящего перехэширования. При расширяющемся хэшировании длина таблицы подвижна - она возрастает или убывает в соответствии с числом записей.

2. Битовые изображения и адресные книги

Пусть у нас имеется m объектов с n признаками, где признак j $(1 \leq j \leq n)$ имеет N_j различных значений. Тогда каждому объекту можно поставить в соответствие битвектор длиной d, где

$$d = \sum_{j=1}^{n} N_j .$$

При этом общий вид элемента битвектора будет следующим

$$q_j^{\kappa} = \begin{cases} 1, & \text{если данный объект имеет значение } \kappa, \kappa=1,\dots,N_j \\ 0, & \text{если данный объект не имеет значения } \kappa, \kappa=1,\dots,N_j. \end{cases}$$

Такой битвектор будем называть битовым изображением объекта. В статье [1] описывается подход, где из битовых изображений создаются адресные книги.

Используя понятия монотонной системы и ядра монотонной системы [2, 3], совокупность битовых изображений B разбивают на группы G_j так, что

$$G_j \subset B, \quad j=1,2,\dots,q, \quad \text{и} \quad G_\kappa \cap G_\iota = \phi,$$

где q – число образуемых групп.

Для каждой группы образуется одно суперизображение, которое будем называть адресом.

В результате рекурсивного использования этой процедуры на совокупности полученных адресов образуется дерево, которое назовем адресной книгой.

Поиск начинается с исследования адреса. Если по адресу видно, что в группе, которую он представляет, нет удовлетворяющих запросу объектов, то вся эта группа дальнейшему исследованию не подлежит. Таким образом, резко уменьшится количество объектов, на которых производится поиск.

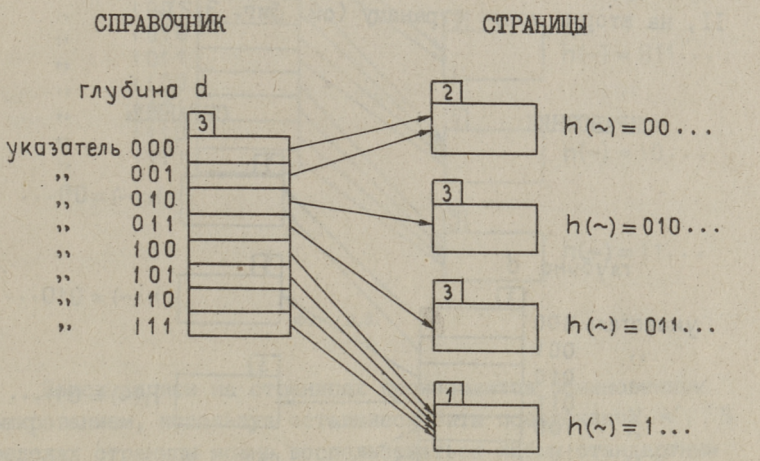
3. Расширяющееся хэширование

При больших наборах данных даже естественные группы могут быть настолько велики, что возникает потребность в отдельном методе поиска записей в пределах группы. Таким методом предлагается расширяющееся хэширование, описанное в статье [5].

Пусть будет задана функция хэширования h и совокупность ключей $K=\{\kappa\}$. Псевдоключом κ' ключа κ назовем величину $\kappa' = h(\kappa)$.

Файл, куда заносятся записи, делится на два уровня – справочник и страницы. Страницы содержат пары $(\kappa, I(\kappa))$, где κ-ключ и $I(\kappa)$ -информация, связанная с этим ключом. Информация может быть либо самой записью, либо указателем на запись.

Справочник имеет заголовок, где хранится глубина справочника d. Кроме заголовка справочник содержит указатели на страницы. Первый указатель отсылает к странице, где записаны ключи, у которых псевдоключи $k' = h(k)$ начинаются d нулями. Затем следует указатель на ключи, имеющие псевдоключи, у которых первые d биты начинаются с 0...0I, тогда указатель на ключи, начинающиеся с 0 ... I0 и т.д. Всего имеется 2^d указателей. При глубине $d = 3$ справочник выглядит, как показано на фигуре 3.I.



Фиг. 3.1.

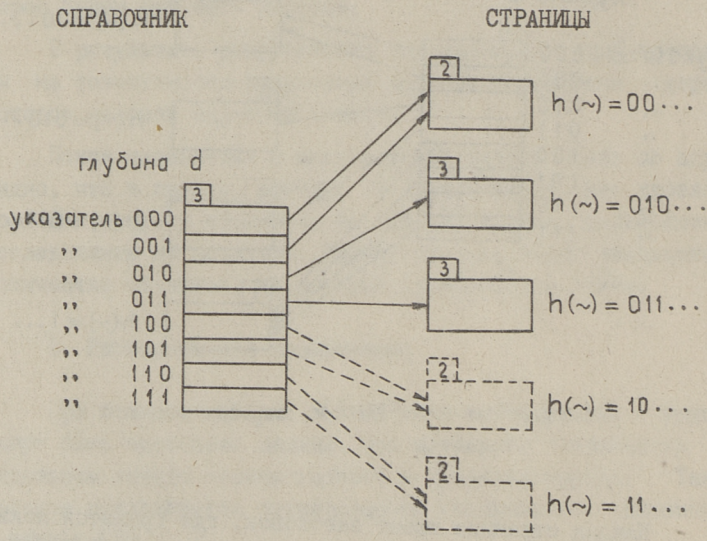
Каждая страница имеет заголовок, где хранится локальная глубина страницы d'. Например, указатель 000 указывает на страницу с локальной глубиной $d' = 2$ (см. фиг. 3.I). Это значит, что на данную страницу записаны все ключи, имеющие псевдоключи, начинающиеся битами 00. В нашем примере на эту страницу указывают 000 и 001. Максимальной локальной глубиной может быть глубина справочника.

Чтобы записать ключ k_0 и связанную с ним информацию, надо вычислить его псевдоключ $h(k_0)$ и найти первые d биты. Затем в справочнике находится соответствующий указатель,

который указывает, на какую страницу надо записать информацию.

Теперь посмотрим, как следует поступить, если нужная страница уже заполнена.

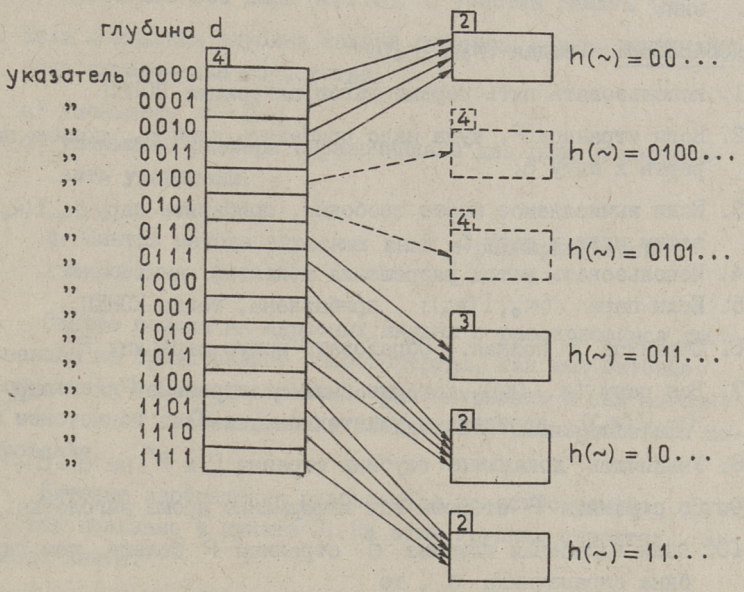
Например, на фигуре 3.I все ключи, у которых псевдоключи начинаются битом I, записаны на одну страницу. Если туда нельзя добавить записей, эту страницу делят на две новые страницы с локальной глубиной 2. Все ключи, у которых псевдоключи начинаются битами I0, записывают на первую страницу, а ключи, у которых псевдоключи начинаются битами II, на вторую новую страницу (см. фиг. 3.2).



Фиг. 3.2.

Если страница полная и ее локальная глубина равна глубине справочника, то последнюю увеличивают на единицу и только затем делят страницу на две.

Например, если нужно прибавить ключ, у которого псевдоключ начинается битами 0I0, а эта страница уже полная, надо действовать, как показано на фигуре 3.3.



Фиг. 3.3.

Поиск записи на страницах производится обыкновенным хэшированием, используя оставшиеся биты псевдоключа k'. В пределах страницы можно воспользоваться любым стандартным методом разрешения коллизий [4]. Алгоритмы для поиска, добавления и удаления записи с использованием расширяющегося хэширования следующие:

ПОИСК (задан ключ k_0)

1. Вычислить $k'_0 = h(k_0)$.
2. Преобразовать первые d биты псевдоключа k'_0 из бинарной формы в целое число и присвоить полученное значение переменной r.
3. С помощью переменной r найти указатель v.
4. По указателю v найти страницу P.
5. С помощью последних d' битов псевдоключа k'_0, где d' - локальная глубина страницы P, вычислить место на странице P, где находится искомая запись.

6. При необходимости использовать метод разрешения коллизий.

ДОБАВЛЕНИЕ (задан $(K_0, I(K_0))$)

1. Использовать пять первых шагов алгоритма ПОИСК.
2. Если страница P, куда надо прибавить ключ K_0, полная, перейти к шагу 6.
3. Если вычисляемое место свободно, прибавить пару $(K_0, I(K_0))$, затем идти к шагу 5.
4. Использовать метод разрешения коллизий.
5. Если пара $(K_0, I(K_0))$ прибавлена, тогда КОНЕЦ.
6. Страница P полная. Образовать новую страницу P^*.
7. Все пары $(K, I(K))$, которые были на странице P, и пару $(K_0, I(K_0))$ записать в участок памяти Q.
8. Увеличить локальные глубины страниц P и P^* на $d' = d' + 1$.
9. Со страницы P стереть всю информацию кроме заголовка.
10. Если локальная глубина d' страницы P больше, чем глубина справочника d, то
    а) увеличить $d = d + 1$,
    б) увеличить справочник в два раза.
11. Добавлять все пары $(K, I(K))$ с участка памяти Q.

УДАЛЕНИЕ (задан K_0)

1. Использовать алгоритм ПОИСК.
2. Если ключа K_0 не найдется, послать соответственное сообщение, и КОНЕЦ.
3. Удалить ключ K_0 в зависимости от метода разрешения коллизий.
4. Если число записей на соседних страницах, у которых $d'-1$ первые биты одинаковые, меньше чем d', действовать следующим образом:
    а) все пары $(K, I(K))$ от обеих страниц записать в участок памяти Q,
    б) удалить одну из этих страниц, поставить указатели удаляемой страницы, указать на оставшуюся страницу P,
    в) уменьшить $d' = d' - 1$,

г) со страницы P стереть всю информацию кроме заголовка,
е) прибавить все пары (K, I(K)) с участка памяти Q.

5. Если локальная глубина каждой страницы меньше, чем глубина справочника d , тогда

а) уменьшить d = d-1,
б) уменьшить размеры справочника в два раза и переставить указатели.

4. Синтез метода адресных книг и расширяющегося хэширования

Синтез поиска на адресных книгах с расширяющимся хэшированием не представляет особого труда, так как битовые изображения можно одновременно рассматривать и как элементы монотонной системы, и как псевдоключи расширяющегося хэширования.

Битовые изображения разбиваются на естественные группы, как показано в пункте 2. На этих группах строится адресная книга.

На каждой группе отдельно проводится расширяющееся хэширование. Поскольку к одной группе относятся однородные битовые изображения, то для хэширования используются биты с тем, чтобы получить равномерное распределение объектов между страницами хэштаблицы.

Поиск начинается с исследования адреса, в ходе которого выясняются группы битовых изображений, подлежащие подробному изучению. В этих группах отыскиваются битовые изображения, удовлетворяющие запросу. Действительные записи выявляются расширяющимся хэшированием.

Такой метод позволяет быстро находить искомые объекты. С помощью адресной книги можно резко уменьшить количество объектов, на которых производится поиск. Кроме того адресная книга уменьшает количество обращений к расширяющемуся хэшированию.

Указанный метод синтеза проходит сейчас экспериментальное апробирование в ВЦ ТПИ.

Литература

I. Выханду Л.К., Выханду П.Л. Быстрый поиск на битматрицах. - Тр. Таллинск. политехн. ин-та, № 554, 1983, с. 49-60.

2. Выханду Л.К. Экспрессметоды анализа данных. - Тр. Таллинск. политехн. ин-та, № 464, 1979, с. 21-37.

3. Кузнецов Е.Н., Мучник И.Б. Анализ распределения функций в организационной системе. - Автоматика и телемеханика. 1982, № 10, с. 119-127.

4. Кнут Д. Искусство программирования для ЭВМ. М., Мир, 1978, том 3, с. 601-615.

5. Fagin R., Nievergelt J., Pippenger N. and Strong H.R. Extendible hashing, a fast access method for dynamic files. - ACM TODS, 4(1979)3, p. 315-344.

L. Vyhandu, P. Vyhandu

How to Synthesize Directory Building and Extendible Hashing

Summary

In this paper a method for quick search using bitmap technique and extendible hashing is introduced. Using monotonic systems theory bitmaps are clustered into "natural" groups to which extendible hashing is applied. Unlike conventional hashing, extendible hashing has a dynamic structure that grows and shrinks gracefully as the number of records grows and shrinks.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 330.115

Э.Х.-Т. Кунапуу

ТЕХНОЛОГИЯ ПРИМЕНЕНИЯ ПАКЕТА ПРОГРАММ СТАТОС ДЛЯ ВЫЯВЛЕНИЯ СОЦИАЛЬНО-ЭКОНОМИЧЕСКИХ ЗАВИСИМОСТЕЙ

Конечной целью создания любых экономико-математических методов является их применение для решения конкретных прикладных задач. В настоящей статье приводится общая технологическая схема использования пакета программ СТАТОС, которая создана на кафедре обработки информации ТПИ, для анализа социально-экономических зависимостей.

Успешное применение экономико-математических методов требует наличия технологии применения. По мнению автора настоящей работы, о технологичности применения экономико-статистического метода можно говорить при соблюдении, по крайней мере, трех условий. Во-первых, метод должен быть реализован на ЭВМ. Во-вторых, должна быть разработана методика использования данного метода, которая охватывает весь путь от планирования исследования до внедрения ее в практику. И в-третьих, необходима документация, отражающая все аспекты использования метода.

I. Выявление зависимостей при помощи разработанного пакета

Одной из причин такого внушительного разрыва между моментом завершения разработки нового математического метода и моментом его внедрения в народнохозяйственную практику является отсутствие методики использования выработанного аппарата. Только тщательно отработанное методическое обеспечение всего процесса исследования может дать высокое качество и надежность результатов.

Исследование социально-экономических зависимостей есть форма научного познания. Процесс получения и использования знаний в теории познания обычно расчленен на три этапа. На первом этапе путем наблюдений, опытов собирают информацию для последующего анализа. На втором этапе на более абстрактном уровне происходит теоретическое осмысление материала, выделение в нем существенного и формируются выводы. На третьем этапе выводы, полученные на основе абстракции, проверяются в практической деятельности. Известно, что данная схема универсальна, хотя в различных областях знания интерпретируется по-разному. В статистических исследованиях данную цепочку теории познания можно интерпретировать следующим образом: изучение и накопление фактов об исследуемой реальной системе - построение и анализ модели системы - использование выводов в практической деятельности.

Проведение конкретных социально-экономических исследований требует, конечно, более детальной схемы. Процесс выявления социально-экономических зависимостей с помощью пакета программ СТАТОС расчленяется на следующие этапы:

I Предмодельный этап: подготовка исследования и предварительный анализ. Здесь,в свою очередь, можно выделить следующие подэтапы:

I.I. Определение цели исследования.

I.2. Анализ исследуемой реальной системы и формирование базы данных.

I.3. Первичная статистическая обработка данных.

П Этап выявления модели зависимости.

П.I. Определение показателей, взаимосвязь между которыми нас интересует.

П.2. Формирование априорных допущений, ограничений и самой модели.

П.3. Статистическое оценивание параметров модели.

П.4. Статистический анализ адекватности модели.

Ш Этап подведения итогов, формирования выводов и практика применения полученных результатов.

Отличительной чертой приведенной схемы исследования является то, что она ориентирована на использование базы данных. Наличие базы данных в пакете придает исследованию но-

вое качество. Процесс исследования становится по-настоящему интерактивным. На каждом шаге анализа исследователь может заказать из базы новые данные, тем самым облегчается реализация цикла научного познания: постановка проблемы - выдвижение гипотезы - проверка гипотезы на данных.

2. Предмодельный этап анализа

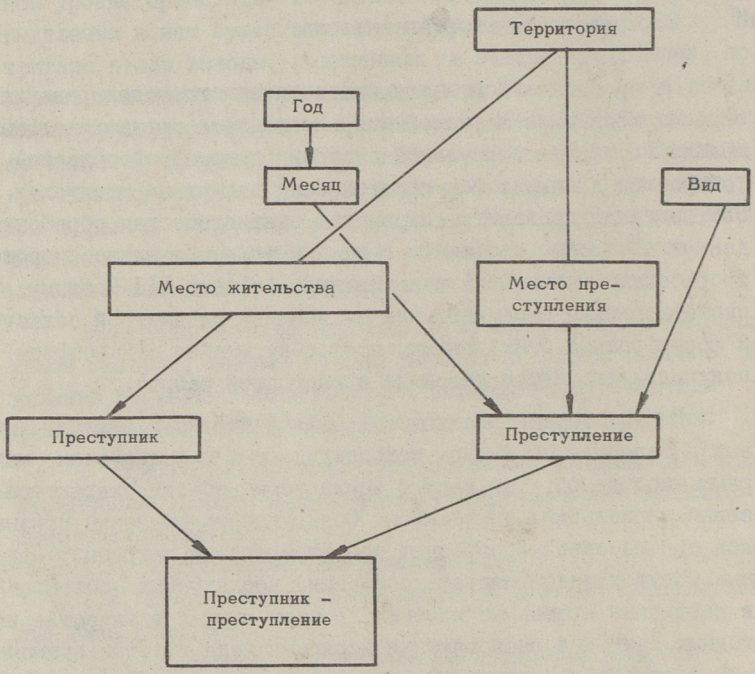
На предмодельном этапе статистического моделирования определяются цели исследования, проводится предварительный анализ исследуемой реальной системы (предметной области), собираются данные и формируется база данных на носителях ЭВМ, а также проводится первичная статистическая обработка данных. Вопросы, связанные с данным этапом анализа, хорошо проработаны и освещены в литературе по теме [I]. Поэтому сконцентрируем свое внимание на анализе предметной области и формировании базы данных, поскольку именно эти вопросы получили дальнейшее развитие в настоящей работе.

В существующих подходах к статистическому анализу данных в качестве структуры исходных данных используются матрицы наблюдений, где каждый исследуемый объект охарактеризован несколькими признаками. Совокупность значений признаков одного объекта образует при этом строку матрицы. Данные могут характеризовать состояние исследуемых объектов в различные временные моменты. В этом случае в качестве исходных данных в распоряжении исследователя имеется временная последовательность вышеописанных матриц.

В настоящей работе принимается более широкая трактовка структуры исходных данных. Предполагается, что исследуемая реальная система может включать несколько взаимосвязанных типов объектов. Такой подход к структурам данных принимается в теории баз данных [2].

Проиллюстрируем данный подход на примере статистического изучения преступности в каком-то регионе. Пример схемы базы данных изображен на фиг. I. На схеме можно выделить три основных информационных объекта. Это "преступник","преступление" и "территориальная единица". Последняя выступает в двух ролях. Во-первых, как место жительства преступника и, во-вторых, как место преступления. Кроме вышеназван-

ных основных информационных объектов на схеме приведены несколько дополнительных. Прежде всего здесь необходимо отметить информационный объект "преступник - преступление", который соединяет конкретного преступника с конкретным преступлением.



Фиг. 1. Схема базы данных для исследования преступности.

При этом предполагают, что в данном преступлении могли участвовать несколько преступников и один преступник мог участвовать в нескольких преступлениях. На схеме изображены еще два дополнительных информационных объекта. Первый классифицирует преступления по видам, второй - по времени совершения преступления.

Разумеется, каждый информационный объект приведенной схемы охарактеризован несколькими признаками. Содержание последних здесь не расшифровывается, поскольку оно ничего не дает для раскрытия идей подхода.

В чем же преимущество приведенной схемы по сравнению с традиционным матричным представлением. Здесь можно выделить следующие характерные моменты:

I. Анализ по приведенной схеме можно проводить, взяв за основу несколько различных объектов. Например, изучаться могут преступники, преступления, территории или виды преступлений.

2. Данная схема очень информативна. Можно получить ответы на вопросы, которые предварительно не были приведены, но возникли в ходе исследования. Например, можно получить характерные данные о месте жительства тех преступников, которые совершают конкретный вид преступления в каком-то конкретном регионе.

Подводя итог, можно сказать, что подход с использованием базы данных дает более гибкие возможности для всестороннего анализа изучаемых явлений.

Рассмотрим конкретнее, как происходит формирование базы данных при использовании пакета программ СТАТОС.

Процесс формирования базы данных можно разделить на три этапа.

I. Анализ предметной области и составление инфологической схемы базы данных. Общие принципы данного процесса изложены в работах [3, 4].

2. Переход от инфологической схемы к конкретной схеме. На этом этапе сравнивают инфологическую схему со стандартными схемами пакета СТАТОС.

По возможности преобразовывают общую инфологическую схему в стандартную. Если это невозможно, то требуется дополнительное настраивание системы. Если же стандартная схема подходит, можно приступить к следующему этапу формирования базы данных.

3. Описание вводных документов и ввод данных в базу. На этом этапе описывают все вводимые данные (тип, "максимальное значение") и подготовляют логические условия проверки начальных данных. Затем происходит непосредственная загрузка базы данных.

После формирования базы данных можно приступать к первичной статистической обработке собранного материала. Общая идеология его проведения и используемые при этом методы изложены С.А. Айвазяном в [I].

Рассмотрим, какими средствами располагает СТАТОС для первичной статистической обработки данных. Методы, реализованные в рамках пакета СТАТОС для первичной обработки данных, можно разделить на следующие группы.

I. Методы образования новых признаков и перекодирования переменных.

2. Методы изучения эмпирических распределений. Для изучения распределений в пакете вычисляются некоторые характеристики распределения (асимметрия, эксцесс, среднее отклонение) и проверяются гипотезы о соответствии выбранной модели распределения исходным данным (по критерию Колмогорова-Смирнова).

3. Методы визуализации данных. Для визуализации одномерных данных используется описание типа "опора и консоль" и "ящик с усами" [5]. Визуализация многомерных данных происходит по методу Эндрюса [5].

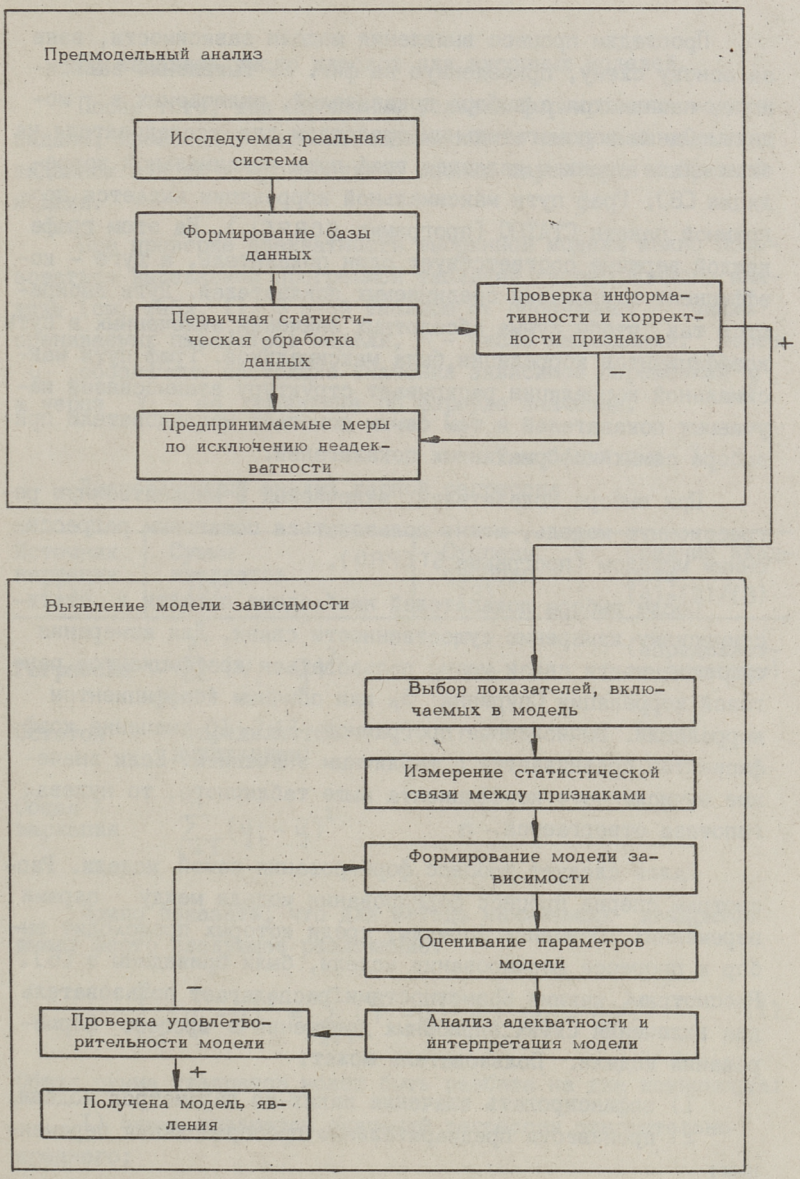
3. Этап выявления модели зависимости

На фиг. 2 приведена общая схема выявления зависимостей с помощью пакета СТАТОС. Рассмотрим, какие возможности имеются в пакете для формирования модели зависимости.

Методы пакета можно разделить на два класса.

I. Методы изучения зависимости между парами переменных. Сюда относятся методика выявления вида функции и методы оценки регрессионных моделей с неизвестными точками структурного изменения [6].

2. Методы оценки моделей с изменяющимися во времени параметрами [7]. Здесь разработаны два алгоритма. Первый - для общего случая (явно-адаптивный метод) и второй - специальный, для оценки динамической производственной функции Кобба-Дугласа.



Фиг. 2. Процесс выявления модели зависимости.

Проследим процесс выявления модели зависимости, взяв за основу схему, приведенную на фиг. 2. Выявление зависимости начинается с выбора показателей, включаемых в модель. Одним вспомогательным средством для исследователя на этом этапе анализа является граф пути максимальной корреляции [8]. Граф пути максимальной корреляции выдается программой пакета СТАТОС (программа CORREL). На этом графе каждой вершине соответствует один показатель, а дуге - коэффициент корреляции соединяемых показателей. Дуги выбираются так, чтобы сумма абсолютных значений, включенных в путь коэффициентов корреляции, была максимальной. Граф пути максимальной корреляции раскрывает структуру взаимосвязей изучаемых показателей и тем самым помогает исследователю при выборе самых информативных показателей.

При выборе показателей, включаемых в множественную регрессионную модель, можно пользоваться пошаговым регрессионным методом (программа STEPRG).

После выбора показателей необходимо перейти к статистическому измерению существенности связи. Для измерения существенности связи можно пользоваться коэффициентом ранговой корреляции Спирмэна S_p или обычным коэффициентом корреляции. Вычисленное программой FNC IO значение коэффициентов сравнивается с табличным значением. Если значение вычисленного коэффициента выше табличного, то нулевая гипотеза отвергается.

Далее следует процесс формирования самой модели. Рассмотрим сперва процесс формирования модели между парами переменных. Основные формулы, среди которых происходит выбор в процессе формирования модели, были приведены в [6]. Рассмотрим, какими возможностями располагает пользователь для включения своих априорных допущений в процесс формирования модели. Пользователь может:

I) зафиксировать значения каких-то параметров модели;

2) произвести предварительные преобразования переменных;

3) зафиксировать классы моделей, в рамках которых ведется поиск;

4) задавать свою собственную модель, которая не включена в класс моделей пакета;

5) задавать число классов для кусочных моделей.

После того, как начальные условия формирования модели заданы, происходит автоматический поиск наилучшей модели в заданном классе и производится оценка параметров выбранной модели.

При проверке адекватности выбранной модели можно пользоваться выдаваемой программой таблицей дисперсионного анализа (см. табл. I). В этой таблице m обозначает число оцениваемых параметров модели, n - число наблюдений. Через y_i обозначены настоящие значения зависимой переменной y, а через Y_i - ее оцененные по формуле значения.

Таблица I

Дисперсионный анализ парной регрессии

Источник вариации (I)	Суммы квадратов (2)	Степень свободы (3)	Средние квадраты (4)=(2)/(3)
Регрессия	$\sum_{i=1}^{n} (Y_i - \bar{y})^2$	$m - 1$	(определяется делением)
Остаток	(определяется вычитанием)	$n - m$	
Общая вариация	$\sum_{i=1}^{n} (y_i - \bar{y})^2$	$n - 1$	-

Можно показать, что для метода наименьших квадратов имеет место следующее разложение:

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (Y_i - \bar{y})^2 + \sum_{i=1}^{n} (y_i - Y_i)^2. \qquad (I)$$

Общая сумма квадратов может быть разбита на два компонента:

I) сумму квадратов значений регрессии относительно среднего;

2) сумму квадратов отклонений относительно линии регрессии (остаточная сумма квадратов).

Если сумма квадратов регрессии будет больше по сравнению с суммой квадратов отклонений относительно линии ре-

грессии, то регрессия будет значимой и в качестве критерия значимости используется

$$F_{m-1,\,n-m} = \frac{\text{средний квадрат регрессии}}{\text{средний квадрат отклонений}} \qquad (2)$$

Вычисленный F-критерий значимости регрессии, который подчиняется F-распределению с $m-1$ и $n-m$ степенями свободы, сравнивается со $100(1-\alpha)\%$-ной табличной точкой $F_{m-1,n-m}$ распределения. Если расчетное значение превышает критическое значение F из таблицы, то нулевая гипотеза отбрасывается.

F-критерий и таблица дисперсионного анализа позволяет судить о том, насколько хорошо линия регрессии приближает действительные данные, но ничего не говорит об адекватности выбранной модели. Настоящая проверка адекватности модели возможна лишь через теоретическое осмысление результатов моделирования. Одним средством, помогающим исследователю на этом этапе анализа, является изучение остатков [6]. Программа пакета СТАТОС выдает графики распределения остатков и соответствующие статистики.

Рассмотрим далее дисперсионный анализ кусочной модели. Изучаемым вопросом в этом случае является вопрос о том, насколько значимо уменьшается общая сумма квадратов при переходе к кусочной модели. В таблице 2 через p обозначено количество кусков, а через n_j - количество наблюдений j-того куска. Для изучения зависимости кусочной регрессии используется отношение уменьшения остатков к общей сумме остатков.

Таблица 2

Дисперсионный анализ. Кусочно-нелинейная регрессия

Источник вариации (1)	Суммы квадратов (2)	Степень свободы (3)	Средние квадраты (4)=(2)/(3)
Одна линия регрессии	$\sum_{i=1}^{n} (Y_i - \bar{y})^2$	$m-1$	
Уменьшение, связанное с переходом к P линиям	(определяется вычитанием)	m_1+m_2-2	
Остаток	$\sum_{j=1}^{P} \sum_{i=1} (y_{ji} - Y_{ji})^2$	$n-m_1-m_2$	
Общая вариация	$\sum_{i=1}^{n} (y_i - \bar{y})^2$		

При выявлении модели с изменяющимися во времени параметрами самым главным является вопрос об устойчивости параметров. Учитывая специфику разработанных в настоящей работе методов, наиболее обоснованной представляется проверка устойчивости параметров на этапе выявления функций изменения параметров. Если на этом этапе выяснится, что распределение какого-то параметра во времени носит случайный характер, то автоматически отвергается гипотеза об изменении данного параметра во времени.

Литература

I. Айвазян С.А., Енюков И.С., Мешалкин Л.Д. Прикладная статистика. Основы моделирования и первичная обработка данных. М., Финансы и статистика, 1983. 471 с.

2. Мартин Дж. Организация баз данных в вычислительных системах. Пер. с английского. М., Мир, 1978. 617 с.

3. Лааст-Лаас Ю.Г. О проектировании логической структуры баз данных экономического анализа. - Тр. Таллинск. политехн. ин-та, 1980, № 482, с. 63-80.

4. Лааст-Лаас Ю.Г. Проблемы инфологического проектирования структур данных для комплекса управленческих задач. - Тр. Таллинск. политехн. ин-та, 1981, № 511, с. 3-14.

5. Выханду Л.К., Нунапуу Э.Х.-Т. Графические методы обработки социально-экономических показателей - Тр. Таллинск. политехн. ин-та, 1981, № 511, с. 101-110.

6. Нунапуу Э.Х.-Т. Автоматическое восстановление монотонных зависимостей по эмпирическим данным. - Тр. Таллинск. политехн. ин-та, 1982, № 524, с. 57-65.

7. Нунапуу Э.Х.-Т. Оценка изменяющихся во времени регрессионных зависимостей. - Тр. Таллинск. политехн. ин-та, 1982, № 524, с. 57-65.

8. Выханду Л.К. Об исследовании многопризнаковых биологических схем. - Применение математических методов в биологии. Л., изд. ЛГУ, 1964, с. 19-21.

E. Čunapuu

How to Use STATOS - a Package for Socio-economical Data Analysis

S u m m a r y

The description of the use of computer package STATOS is given. The package provides tools for data analysis process starting from data base creation up to the statistical model building.

№ 568
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.06

Я.Я. Тепанди

ГЕНЕРАЦИЯ ОТЧЕТОВ СТАТИСТИЧЕСКОГО ТИПА В СРЕДЕ СУБД

I. Введение

Разноцелевые проблемно-ориентированные системы обработки данных (ПО СОД) находят все большее применение во всех отраслях народного хозяйства. Особенно целесообразно создание систем с несложными входными языками, освоение которых под силу специалисту предметной области. К таковым относятся ПО СОД, создаваемые с помощью инструментальной системы СХОДИ [I]. Основные функции этих систем - (I) ввод, обновление и реорганизация данных, (2) составление отчетов статистического и информационно-поискового типа. Созданные ПО СОД являются надстройками сетевой СУБД.

Отчет статистического типа - это таблица, для которой имеется общее логическое условие (ЛУ) отбора данных, а также ЛУ для каждой строки и каждого столбца [2]. Роль СУБД при составлении отчетов в СХОДИ определяется, в основном, тем, что с ее помощью отбираются записи для генератора отчетов. В каждую клетку отчета суммируется число записей, удовлетворяющих одновременно общему условию, условиям соответствующей строки и соответствующего столбца.

В настоящей статье основное внимание уделяется развитию методов составления отчетов статистического типа, с целью более полного использования возможностей СУБД.

2. Язык описания статистических отчетов

Требования к языку. В ходе эксплуатации созданных ПО СОД выяснилось, что целесообразно было бы иметь следующие возможности, кроме уже имеющихся:

- возможность определить число строк (столбцов) и оформление отчета, зависящее от содержания базы данных;

- возможность определить иерархические отчеты, в которых как строки, так и столбцы организованы в многоуровневую структуру (отметим, что в СХОДИ во время печати можно имитировать внешний вид иерархических отчетов [3], на уровне же описания структуры таблицы такое понятие отсутствует - например, ЛУ строки задается как конъюнкция ЛУ всех верхних уровней этой строки);

- возможность провести арифметические действия над группами строк и столбцов (агрегирование арифметических действий над строками и столбцами);

- возможность определить перед составлением отчета переменные, которые входят как в оформление (тексты), так и в условия таблицы.

В предлагаемом ниже языке эти требования учтены.

Внешняя форма языка. Предлагаемый язык имеет линейную форму. Запрос (программа) на этом языке записывается в виде синтаксически организованной последовательности конструкций, а не задается в ходе диалога, с помощью графических средств и т.д. Для некоторых типов систем (например, когда выходные формы образуются графически на дисплее) такой язык может показаться лишним. Мы все же убеждены, что линейная форма языка выполняет по меньшей мере две важных функции: (а) его синтаксис служит для формального описания функций систем и возможностей пользователя, и (б) программа на этом языке служит основой для изменений и повторного использования запросов.

Например, в случае диалоговой системы целесообразно определить функции системы с помощью синтаксиса и семантики промежуточного линейного языка. Это, однако, не означает, что промежуточный язык должен реализоваться в системе: диалог может сразу интерпретироваться. Часто, наоборот, линейная форма языка является основной, т.е. строят (или применяют) систему с линейным входным языком и над этим языком надстраивают диалоговый или графический интерфейс.

Выбор СУБД. Рассмотрим для конкретности язык определения отчетов (ЯОО) в среде реляционной СУБД SEQUEL 2 [4].

Выбор указанного языка продиктован следующими соображениями. Это один из наиболее легко усваиваемых и мощных языков, имеющих линейное представление; программы на этом языке относительно короткие, а соответствующая СУБД в настоящее время усиленно разрабатывается фирмой IBM [5]. Средства, аналогичные используемым ниже, имеются, однако, во многих других языках. Поэтому ценой часто небольших изменений ЯОО представим в среде языка манипулирования данными (ЯМД) других СУБД.

В примерах данной статьи используется фрагмент БД, состоящий из следующих двух отношений:

СЛУЖ(СЛНО, ИМЯ, ОТНОМ, ЗАРАБ),
ОТДЕЛ(ОТНОМ, ОТИМЯ, РАСПОЛ).

Принципы построения ЯОО.

I. ЯОО совмещается с ЯМД, использует все его возможности, сохраняет по возможности стиль ЯМД.

2. Во избежание конфликтов понимания в ЯОО не используются конструкции с "незначительно" отличающейся от ЯМД семантикой. В сомнительных случаях вводятся новые конструкции.

3. Вводится понятие отношения с текстом: это таблица, к определенным столбцам которой прибавляются постоянные тексты. Для определения отношения с текстом можно использовать конструкцию SELECT языка SEQUEL 2. Например, запрос

```
SELECT ('ОТДЕЛ', ОТИМЯ, 'НАХОДИТСЯ'), ('В', РАСПОЛ)
FROM   ОТДЕЛ
```

создает отношение, состоящее из пар

(ОТДЕЛ ...НАХОДИТСЯ, В .....).

Отметим, что обычно добавление постоянных текстов используется при оформлении выходных данных. Здесь это средство введено на уровне ЯМД для описания отчетов (это - тоже выходные данные). Отметим также, что прибавление постоянных текстов можно рассматривать как операцию произведения в реляционной алгебре. Результирующее отношение не находится в третьей нормальной форме.

4. Тексты строк и столбцов определяются как отношения с текстами. Точнее, каждый уровень описания строк или столбцов иерархического отчета определяется как отношение с текстом.

5. Сам отчет является результатом взаимного влияния (интерферирования) двух иерархий отношений: одной для строк, другой - для столбцов.

6. Шапки и подписи к отчетам могут представлять собой отношения (с текстами). В таком случае нужные действия повторяются для всех кортежей отношений, например, выдается множество таблиц.

7. Внешнее оформление и условия заполнения отчетов могут задаваться раздельно (преимущества - лучшая структуризация ЯОО, таблицу можно преобразовать перед оформлением), или вместе (преимущества - исключается дублирование описания структуры таблицы; легче переход к графическому вводу запроса). Имеются средства преобразования запроса от одного представления к другому.

8. В запросе на отчет могут задаваться параметры, которые могут одновременно использоваться при отборе данных, при определении содержания и текстов отчета.

9. В язык включены средства документирования программ ЯОО: комментарии, мнемонические ключевые слова, вариант ЯОО с документирующей лексикой (длинные ключевые слова), средства перехода от одной лексики ЯОО к другой.

10. В язык включены средства хранения и повторного использования запросов.

3. Компоненты ЯОО

Типичный запрос на отчет состоит из следующих компонентов:

- отбор данных;
- преобразование данных;
- описание содержания таблицы (иерархический или неиерархической);
- преобразование таблицы;
- оформление таблицы;

- вывод таблицы в оформленном или неоформленном виде.

Рассмотрим соответствующие компоненты ЯОО в среде ЯМД СУБД. Основное внимание уделяется описанию содержания и оформления таблицы.

Отбор данных из БД. В системах с языком манипулирования данными высокого уровня, например, в реляционных СУБД, отбор осуществляется средствами самой СУБД. В системах с языком более низкого уровня для этого может потребоваться надстройка над СУБД, облегчающая пользователю составление запроса на отбор. Связь с данной формой устанавливается одним из следующих способов:

- запрос на отбор включается в запрос на форму;
- в запрос на форму включается имя отношения, которое либо существует, либо создается как пользовательское (DEFINE VIEW),
- запрос на отбор задается при вызове запроса на форму.

Преобразования над отобранными данными производятся средствами СУБД. Синонимы задаются в предложении DEFINE VIEW. При описании больших таблиц может оказаться целесообразным присвоение имени атрибута по умолчанию синонима, являющегося номером атрибута в списке атрибутов отношения. Например, если определено отношение

```
DEFINE VIEW    ОТД (ОТДЕЛЕНИЕ, ЗАРП) AS
SELECT         ОТНОМ, ЗАРАБ
FROM ...,
```

то условия:

ЗАРП > 200 и 2 > 200

имеют одинаковое содержание.

Арифметические операции и образование новых атрибутов реализуется либо средствами СУБД в предложении отбора данных, либо непосредственно в описании отчета. Может оказаться целесообразным "ленивое" выполнение преобразования, как, например, в следующем условии:

(ЗАРПЛАТА > 1000) & (СРЕДНЯЯ)СТАРЗАРПЛ) > 500).

Если первое условие не выполняется, то вычислять второе уже нет необходимости. Если вычисления СРЕДНЯЯ(СТАРЗАРПЛ) не производить до того момента, когда это

действительно необходимо, можно добиться значительной экономии времени [6].

Преобразования таблицы - арифметические действия над строками и столбцами, удаление или добавление строк или столбцов, действия между разными таблицами - требуют обычно специального программного обеспечения. Если рассматривать таблицу отчета как отношение реляционной БД, то некоторые из этих преобразований выполняются средствами СУБД.

Описание содержания и оформления (текстов шапки, боковин, подписей) таблицы рассматривается ниже вместе.

Описание неиерархического отчета. Исходим из принципа, изложенного выше, по которому таблица получается как результат взаимовлияния отношений для строк и столбцов. Отношение для строк может рассматриваться как таблица с одним столбцом. Зафиксируем данные, по которым производилось вычисление одной строки такой таблицы, (т.е. одного элемента), а также способ вычисления. Зададим теперь ограничения на данные, а, возможно, и изменения в способе вычисления элемента. Если эти ограничения одинаковы для всех строк, то тем самым определен новый столбец отчета. Следует отметить, что ограничения на данные можно наложить почти произвольно, но изменения в способе вычисления элементов должны быть согласованы. Например, если в строке вычисляется сумма значений одного атрибута, а в столбце - другого атрибута, то результат может быть неопределенным. Симметрично, можно начать с отношений для столбцов (таблицы в одну строку) и описать строки. Таким образом, нужны средства описания таблицы в одну строку или в один столбец.

Для описания такой таблицы нужно задавать способ вычисления каждого элемента. Можно ли это сделать средствами СУБД? Иногда, да. Например, в SEQUEL 2 имеется функция COUNT подсчета числа кортежей, удовлетворяющих данным ограничениям. Таким образом, в этом языке можно описать таблицу в один столбец определенного вида, хотя это и длинно. Например, для описания одной строки (отделение 30) такой формы мог бы понадобиться фрагмент:

```
SELECT COUNT (*)
FROM   СЛУЖ
WHERE  ОТНОМ = 30.
```

Далее, существует средство GROUP BY, осуществляющее группировку кортежей, имеющих одинаковое значение некоторого атрибута. Это укорачивает описание таблицы. Тексты строк и столбцов можно задавать как отношения с текстом. Таким образом, имеются все нужные средства: используя последовательность предложений SELECT в сочетании с предложением GROUP BY и функциями COUNT, SUM и другими, можно описать таблицы в один столбец или в одну строку.

Возникает, однако, интересный вопрос: правильно ли истолкованы в этом случае предложения ЯМД? Ведь, в данном случае, предложение SELECT будет обозначать уже не только образование нового отношения, но и непроцедурно заданное предписание образования матричной формы.

Ответ на этот вопрос, вероятно, отрицательный. В данном случае возникает смещение семантики ключевых слов: некоторой конструкции присваивается семантика, немного отличная от первоначальной. Это увеличивает вероятность ошибок. Лучше сохранить за предложениями ЯМД только их старую семантику, а для обозначения новых действий ввести и новые конструкции. Подчеркиваем, что этот вывод относится только к синтаксическому определению отчета. Здесь мы говорим "в терминах предметной области": вместо определения "отношения с текстом" определим "отчет в один столбец". Рассматривая же отчет как абстрактный объект, целесообразно определить его как результат взаимодействия двух отношений.

Учитывая вышесказанное, строки и столбцы отчета целесообразно задавать через самостоятельные ключевые слова (ROW, COL). Описание строки (столбца) может включить текст, описание данных и описание действий. Например, запись

```
ROW
SELECT    'ОТДЕЛЕНИЕ 30', SUM (ЗАРАБ)
FROM      СЛУЖ
WHERE     ОТНО = 30
```

описывает одну строку отчета в один столбец. Строка снабжена текстом ОТДЕЛЕНИЕ 30, и в нее суммируется заработок служащих 30-го отделения. Таблица в один столбец определена последовательностью таких конструкций.

Группа строк или столбцов может задаваться с использованием конструкции GROUP BY. Пример: описать последова-

тельность строк. Для каждого отделения задается своя строка. Текст к-той строки "ПЕРСОНАЛ К-ГО ОТДЕЛЕНИЯ", и в нее вписывается число служащих к-го отделения.

```
ROW
SELECT  (' ПЕРСОНАЛ ', ОТНОМ,' -ГО ОТДЕЛЕНИЯ'), COUNT (*)
FROM    СЛУЖ
GROUP BY  ОТНОМ.
```

Та же задача может решаться с использованием вспомогательной переменной, пробегающей все значения из данного отношения:

```
FOR X IN (SELECT  ОТНОМ FROM ОТДЕЛ)
ROW
SELECT  ('ПЕРСОНАЛ',  X ,' - ГО ОТДЕЛЕНИЯ), COUNT (*)
FROM  СЛУЖ
WHERE  ОТНОМ = X
```

Отметим, что эти решения неэквивалентны. В последнем случае создаются строки и для отделов без служащих, в первом - не создаются. Запрос второго вида, полностью эквивалентный первому, легко записывается. Нужно только первую строку заменить на

```
FOR X IN (SELECT UNIQUE  ОТНОМ  FROM  СЛУЖ).
```

Записать же запрос первого вида, эквивалентный второму, немного труднее. Это тем труднее, если имеется множество переменных, пробегающих каждое свое отношение.

В случае иерархических таблиц желательно каждым предложением ЯМД описывать только одну строку отчета в один столбец (или наоборот), а множество строк задавать с помощью внешних операций (FOR ... IN ...). Используем еще операторные скобки BEGIN ... END и допускаем вложенность конструкций ROW, COL. На каждом уровне иерархии строк или столбцов определим таблицу в одну строку или в один столбец. Для каждого столбца (каждой строки) такой таблицы определим новую таблицу в одну строку или один столбец и т.д. Естественное правило при этом состоит в том, что при определении более низких уровней иерархии мы можем сузить ограничения, с которыми имеем дело, но не расширить их. Это условие можно характеризовать как ЕСТЬ-иерархию в концептуальной схеме [7]: представитель более низкой иерархии унаследует все свойства высшей иерархии, и приобретет вдоба-

вок новые свойства. Как и в концептуальной схеме, было бы разумно не переносить описание "отца" в описание "сына". В следующем примере, однако, это делается, чтобы подчеркнуть наличие отношений на каждом уровне иерархии.

Пример. Описать столбцы формы, имеющей следующую шапку (в клетки записывается число служащих в городе, селе, Таллине и т.д.):

ГОРОД		
ТАЛЛИН	ОСТАЛЬНЫЕ	СЕЛО

```
COL BEGIN
 SELECT    'ГОРОД',* FROM  СЛУЖ WHERE ОТНОМ  IN
 SELECT    ОТНОМ  FROM ОТДЕЛ  WHERE РАСПОЛ ¬= 'СЕЛО' .
 COL SELECT ТАЛЛИН',  COUNT(*) FROM  СЛУЖ WHERE ОТНОМ IN
 SELECT     ОТНОМ  FROM ОТДЕЛ  WHERE  РАСПОЛ = 'ТАЛЛИН'
 COL SELECT 'ОСТАЛЬНЫЕ', COUNT(*) FROM  СЛУЖ  WHERE ОТНОМ IN
 SELECT    ОТНОМ  FROM ОТДЕЛ
       WHERE  РАСПОЛ ¬= 'СЕЛО' AND РАСПОЛ ¬= 'ТАЛЛИН'
END
COL SELECT 'СЕЛО', COUNT(*) FROM   СЛУЖ  WHERE ОТНОМ  IN
    SELECT ОТНОМ  FROM ОТДЕЛ  WHERE  РАСПОЛ = 'СЕЛО'
```

Упрощение запроса. Предыдущий пример показывает, что желательно укоротить запрос. Это можно сделать, если:

- перенести на низкие иерархии только ограничения;
- образовать новые отношения и дать им имена;
- опустить, где можно, ключевые слова ЯМД СУБД;
- задать определенные действия (например, COUNT) по умолчанию;
- ввести новые ключевые слова ЯОО;
- использовать более простые имена таблиц и атрибутов.

При этом предыдущий запрос мог бы приобрести следующий вид (предположим, что при запросе на отбор данных определено отношение, первый атрибут которого - номер служащего, а второй - расположение отделения).

```
COL (2 ¬= 'СЕЛО',      TEXT      'ГОРОД'
    COL  2 = 'ТАЛЛИН',  TEXT      'ТАЛЛИН'
    COL 2 ¬= 'ТАЛЛИН', TEXT  'ОСТАЛЬНЫЕ')
```

COL 2 = 'СЕЛО' TEXT 'СЕЛО'.

Вводя такие соглашения, мы все дальше отходим от ЯМД СУБД и получаем проблемно-ориентированные языки обработки данных различной сложности и мощности. При этом необходимо иметь и соблюдать вышеприведенные принципы, чтобы не вступить в противоречие с логикой СУБД.

Агрегация строк и столбцов, например, вычисление сумм строк, можно описать:

- в запросе на форму, с помощью условий отбора в данную строку (столбец);
- в запросе на форму, с помощью специальных прерываний (например, если оканчиваются строки, относящиеся к некоторому району, вычисляется новая строка - их сумма);
- после вычисления таблицы, с помощью специального подъязыка преобразования таблицы.

Независимость от СУБД. На самом деле, пользователя интересует не множество ЯОО, предложенных выше, а один язык, который может надстраиваться над разными СУБД. В этом случае легче перейти с одной СУБД на другую, с одной ЭВМ на другую. Для этой цели можно:

- взять в основу конкретную СУБД, а при переходе на другую СУБД перевести описания данных на ЯМД последней(или изобрести собственный ЯМД и перевести его в ЯМД выбранной СУБД, что почти то же);
- минимизировать появление операторов ЯМД, локализовать их, строго определить места их появления, определить интерфейсы между ЯМД и ЯОО (например, допускать в ЯОО только имена отношений или файлов);
- совершить комбинированный подход - допускается свободное использование конструкций некоторого ЯМД, а в случае других ЯМД ставятся ограничения.

Принципы, изложенные в статье, не ограничивают применения указанных подходов. Например, в первом случае используем в ЯОО конструкцию GROUP BY, во втором - FOR ... IN.

Заключение. В статье предложен ЯОО, функционирующий в среде ЯМД реляционной СУБД. Предложенные принципы построения ЯОО могут распространяться на ЯМД СУБД разного типа,

так как отчет в одну строку (один столбец) - простейшая структура, которую можно образовать средствами всех СУБД.

Литература

I. Выханду Л.К., Микли Т.И., Тепанди Я.Я. О технологии построения проблемно-ориентированных систем обработки данных. - Тр. Таллинск. политехн. ин-та, 1983, № 554, с. 13-19.

2. Аус Т.А., Рябовыйтра М.Г., Томбак М.О. Генератор матричных отчетов. - Тр. вычислительного центра Тартуск. гос. ун-та, 1974, № 30, с. 23-30.

3. Бернштейн Е.Б. Средства печати таблиц в системе СХОДИ. - Тр. Таллинск. политехн. ин-та, 1982,№ 524, с. 39-49.

4. Chamberlin D.D. et al. Sequel 2: A unified approach to data definition, manipulation, and control. - IBM Journal of Research and Development, November 1976, vol. 20, N 6, p. 560-575.

5. Jaakkola H. Kyselykielen Query D Suunnittelu System D - järjestelmää varten. - Matemaattisten tieteiden laitos, Tampereen yliopisto, raportti A91. Lokakuu 1982, 141 s.

6. Тепанди Я.Я. О проверке данных в проблемно-ориентированных пакетах прикладных программ. - Тр. Таллинск. политехн. ин-та, 1981, № 511, с. 15-24.

7. Roussopoulos N. CSDL: a conceptual schema definition language for the design of data base applications. - IEEE Trans. on Software Engineering, September 1979, vol. SE - 5, N 5.

J. Tepandi

Generation of the Statistical Accounts in DBMS Environment

S u m m a r y

In this paper the generation of statistical accounts the contents of which is determined by two groups of conditions, one for rows and the other for columns, is considered.

The main problem is, how to use DBMS means to describe the accounts. The general principles are given. The SEQUEL 2 is used as an example DBMS.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.06

Я.Я. Тепанди

КРИТЕРИИ ОЦЕНКИ ЯЗЫКОВ С ОБЩИМ ЯДРОМ

I. Введение

Структурное программирование совершило переворот во взглядах на качество программ. По разным, практически обоснованным, причинам одними из главных достоинств программ стали считаться их ясность и простота. Во многих источниках даются рекомендации по написанию таких программ, устанавливаются критерии их оценки (см. например, [I]). Это - одна возможность выполнения крайне актуальной задачи улучшения качества программного обеспечения.

Другая возможность - это использование языков другого типа, например, проблемно-ориентированных языков [2]. Последние позволяют в сжатом виде описывать задачи предметной области данного пакета программ. В сжатом виде - понятие относительное. Если задачи сложные, то нужно либо ограничивать предметную область (задачи становятся однотипными, их описание укорачивается), либо написать "сжатые" описания в несколько страниц. Короче говоря, простота и ясность программ, написанных на проблемно-ориентированном языке (ПОЯ), так же важны, как и простота и ясность программ, написанных на универсальных языках.

Известно, что некоторые языки способствуют написанию более качественных программ, чем другие. Анализ качества языков и программ может идти по нескольким направлениям: анализ программных метрических характеристик [3], анализ таких человеческих факторов, как скорость обучения и количество ошибок в программах [4], сравнение объемов программ и длительностей их составления для разных языков [5], определение характеристик программы через формальные критерии [6].

В данной статье делается попытка дать критерии лаконичности и понятности ПОЯ. Эти понятия тесно связаны с понятиями, соответственно, простоты и ясности ПОЯ, а в конечном счете - с понятием качества ПОЯ. Рассматриваемые языки - это так называемые языки с общим ядром. Общее ядро, в данном случае - это входной язык пакета программ, а ПОЯ возникают как результат заполнения "пробелов" в программах языка-ядра. В зависимости от числа и широты таких "пробелов" возникают ПОЯ с разными свойствами.

2. Пример проблемно-ориентированных языков с общим ядром

В данном разделе приводится пример языка-ядра и образованных на его основе ПОЯ. В качестве языка-ядра рассматривается входной язык системы СХОДИ [5]. Пользователь описывает свои задачи на этом языке во время генерации своей системы. В составленных программах оставляются "пробелы", незаполненные места, которые доопределяются во время запуска программы. Расположение таких "пробелов" в программе, а также их протяженность полностью задается пользователем. Текст, который вставляется вместо "пробелов", является рассматриваемым в данной статье ПОЯ (точнее, программой на ПОЯ). В зависимости от расположения и протяженности "пробелов" возникают более или менее понятные или лаконичные ПОЯ. Поясним эти понятия подробнее [7].

Средства переключения каналов. Эти средства дают возможность настройки системы на текущие задачи. В качестве примера можно привести задачи вычисления отчета некоторой формы в разных разрезах (по месяцам, районам и т.д.). Первая такая возможность реализуется с помощью символа динамической модификации (СДМ) запроса (символа #), после которого пишется новый номер N канала ввода. Следующие за этим номером символы текущей записи (перфокарты) не учитываются, продолжение программы считывается уже по новому каналу. Символ # на синтаксический уровень не выносится. Запрос начинает считываться по каналу N.

Пример. Пусть под DD-именем FT05F001 в языке управления заданиями для программ системы СХОДИ задается раз-

дел библиотеки L629.REPSOU(F28), в котором хранится текст основного запроса:

```
PRO=SYN, TEXT=#10
*/K1N15*
```

(условия для строк и столбцов,
составление и печать отчета)

```
@@
```

Пусть после оператора языка управления заданиями

```
FT10F001 DD *
```

имеется перфокарта модификации запроса:

```
K18E'ЕРЕВАН'&16E80#5
```

Тогда первая карта программы берется по каналу 5, вторая по каналу I0, третья - опять по каналу 5. В результате выполняется программа

```
PRO=SYN, TEXT=
K18E'ЕРЕВАН'&16E80*/K1N15*
```

(условия для строк и столбцов,
составление и печать отчета)

```
@@
```

Если I8-й признак указывает на место происшествия (город), а I6-й - на год, то теперь таблица вычисляется в разрезе города Еревана на I980-й год. Такое переключение канала ввода может происходить в произвольном месте программы и в произвольное число раз, что дает возможность произвольно модифицировать текст основного запроса. Переключение каналов может рассматриваться как простейший макрогенератор (в текст основного макроопределения вставляются параметры).

Оператор переключения каналов расширяет возможности символа переключения каналов, описанного в предыдущем абзаце. Возможно задание фортрановских номеров канала ввода запроса, канала вывода таблицы и т.д.

Недостаток оператора переключения каналов по сравнению с символом переключения каналов проявляется в том, что оператор дает возможность перехода на новый канал только между двумя другими операторами, а символ # - в произвольном месте. С другой стороны, символ # изменяет только номер канала ввода запроса.

Реализация переключения канала. Система СХОДИ реализована на Фортране. С этого языка имеются хорошо отработан-
н

ные оптимизирующие трансляторы, поэтому программы на Фортране по эффективности почти не уступают программам, написанным на Ассемблере. Однако обработка символьной информации на Фортране неудобна. В [I] предложена подпрограмма-функция GETC, осуществляющая ввод символьной информации с внешнего носителя. В подпрограмме объявлен буфер, куда считывается запись с символьной информацией. Каждое обращение к подпрограмме вызывает выдачу следующего символа. По окончании записи считывается новая запись. Если файл исчерпан, выдается резервированное значение конца файла.

Описанные идеи используются и в модуле ввода символьной информации в СХОДИ. Однако в этот модуль внесены следующие существенные дополнения, которые превращают программу в эффективное средство настройки:

- номер после СДМ означает, что следующая запись считывается с фортрановского канала под этим номером, т.е. происходит переключение канала ввода. Операционная функция такого переключения описана выше;

- введена возможность копирования вводимых записей в целях отладки программ пользователя;

- введен символ перехода к новой записи, используемый при вводе форматов языка Фортран при распечатке отчетов.

Ниже приводится алгоритм подпрограммы-функции системы СХОДИ.

Алгоритм А. Посимвольный ввод с переключением каналов.

Вход:

- номера каналов К1 (ввод запроса, по умолчанию К1=5), К2 (копирование запроса, по умолчанию К2=6);
- признак копирования запроса ПКЗ (по умолчанию, ПКЗ=0);
- коды служебных символов С1 (конец оператора, по умолчанию - @), С2 (конец записи, по умолчанию - ↓), СДМ;
- резервированные выходные значения РЗ1, РЗ2 функции GETC, выдаваемые соответственно в ответ на символ С1 и при исчерпывании файла по каналу К1;
- вводимый запрос по каналу К1;
- длина вводимой записи ДЛИНАЗАП (по умолчанию - 72);
- исходное значение счетчика символов СЧС (равно ДЛИНАЗАП).

Значения по умолчанию для переменных ПКЗ, К1, К2, С1, С2, СДМ, Р31, Р32 передаются через общую область.

Выход

Каждое обращение вида К = GETC(С) вызывает присвоение переменным К и С кода очередного вводимого символа. Специальные символы вызывают действия, описанные выше. Возможно копирование запроса по каналу К2.

Метод

```
Начало
  нач: СЧС = СЧС + 1;
       если  СЧС > ДЛИНАЗАП то
       коммент запись выдана, читать новую запись;
            начало
                 читать файл К1 в БУФЕР;
                 если (конец файла), то
                      начало С = Р32; идти к возв; конец
                 если ПКЗ = 1 записать БУФЕР в файл К2;
                 СЧС = 1;
            конец;
       ветвление по БУФЕР(СЧС) начало
            коммент конец оператора;
            ветвь (С1)  С = Р31;
            коммент признак конца записи;
            ветвь (С2) начало СЧС = ДЛИНАЗАП; идти к нач;
            конец
            коммент символ переключения канала;
            ветвь (СДМ) начало

                        если СЧС = ДЛИНАЗАП то идти к нач;
                        проверить, являются ли цифрами первый
                        или первые два символа, непосредственно
                        следующие за СДМ в пределах записи. Если
                        да, преобразовывать их в десятичное число и
                        присвоить К1;
                        СЧС = ДЛИНАЗАП;
                        идти к нач; конец;
            коммент так как СЧС теперь равен ДЛИНАЗАП, будет
            считываться новая запись, притом по каналу
            с новым номером К1. Неправильные символы после
```

```
            СДМ игнорируются (можно выдать сообщение об
            ошибке);
            коммент обычные символы передаются вызываю-
            щей программе;
            иначе  С = БУФЕР (СЧС);
        конец ветвления
возв: GETC = C

        возврат
конец
```

Модификация алгоритма А. В алгоритм А нетрудно ввести дополнения, в результате которых отпадает необходимость в номере канала (номер нового канала задается по умолчанию) в переходе на новую запись при получении СДМ (запоминаются два указателя - для записей основного и вспомогательного файлов ввода) и во внесении СДМ в конец запроса (по окончании вспомогательного файла происходит автоматический возврат на основной запрос). Приведенные ниже примеры основываются на таком алгоритме.

Свойства языка настройки. Пользователь свободен в выборе фрагментов запроса, которые задаются при выполнении запроса с помощью СДМ. Совокупность этих фрагментов образует ПОЯ настройки данной задачи. Например, в приведенном выше примере образуется язык, который можно описать с помощью следующих правил (с учетом модифицированного алгоритма):

```
<программа>  :: = К I8 Е' <город> '& I6 Е  <год>
<город>  :: = <буква> | <город> <буква>
<год> :: = <цифра>  <цифра> .
```

Заметим, что в запросе содержится много "лишней" информации: последовательность букв К I8 Е', с которой начинается запрос, повторяется в каждом запросе и ее можно было бы отбросить. Действительно, вставляя СДМ после символа ' (а не после символа =, как в примере), мы перенесем начало программы на ПОЯ в основной запрос. Мы можем еще ввести два переключения канала, получая следующий язык:

```
<программа I> :: = <город> # <год>
```

Запрос на программу задается теперь уже в виде

```
ЕРЕВАН # 80
```

Далее, мы можем закодировать названия городов и годов, получая таким образом все более короткие запросы, например, 3 # 0, где 3 - код города Еревана, а 0 - код года 80.

Программы становятся короче, не содержат лишней информации. Мы можем сказать, что язык становится лаконичнее. В то же время программы становятся все более непонятными - сравниваем между собой, например, два запроса:

форма 4.5 # РАЙОН = I5 # БОЛЬНИЦА = 30,
4.5 # I5 # 30.

Можно заметить, что (обычно) чем больше контекста переносится в ПОЯ, тем понятнее он становится.

Практическое использование перемены канала. Перемена канала используется в половине внедренных ПО СОД (в более, чем 600 программ запросов, главным образом в выходных формах). Время, затрачиваемое на работу алгоритма, составляет ничтожную долю времени работы всех программ. Результаты опроса показывают, что пользователи предпочитают лаконичную форму запроса. Причина ясна - никто не хочет делать лишнюю работу. Аналогичное явление наблюдается в программировании на алгоритмических языках - программисты не слишком соблюдают дисциплину, например, структурного программирования на Фортране. Однако в лаконичных программах легче допускаются содержательные ошибки (хотя, ввиду краткости программ, синтаксических ошибок может быть меньше). Этот недостаток можно частично устранить с помощью организационных мер: тщательно подготовленные руководства оператора, хорошая подготовка программистов-параметристов и т.д. Этим тоже создается своеобразный контекст, превращающий запрос в более понятный. Только этот контекст теперь не включается в программы, а находится в сознании людей, в руководствах и т.д. Учет этого контекста требует более общего подхода к ПОЯ.

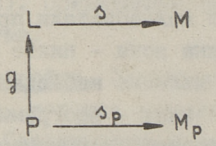
3. ЛАКОНИЧНОСТЬ И ПОНЯТНОСТЬ ПОЯ

Относительность понятий. Мы хотим дать критерии лаконичности и понятности, т.е. измерить их. Измерить объект - это значит сравнить его с чем-либо внешним. Что является внешним к ПОЯ? Пользователь, ЭВМ, операционная система, пакет программ, родной язык пользователя или еще что-нибудь?

По-видимому это все и еще многое другое. В зависимости от нашего выбора мы получаем различные меры и критерии оценки наших понятий.

Языки, значения, функции. Пусть $\Sigma = \{a_1, a_2, \ldots, a_n\}$ - конечный алфавит, а L - язык над алфавитом Σ, т.е. $L \subseteq \Sigma^*$. Пусть M - множество значений для слов из L, а δ - функция семантики, сопоставляющая каждой $y \in L$ некоторое единственное значение $m \in M$. Элемент множества M может быть, например, целым числом, сообщением типа "синтаксическая ошибка", деревом разбора цепочки y и т.д.

Пусть P - ПОЯ над алфавитом Σ, а g - функция, сопоставляющая каждой x из P однозначно определенную y из L (интуитивно, g - обратная функция функции, генерирующей P). Пусть M_p - множество значений для P, а δ_p - функция семантики из P в M_p. Получаем следующую картину (фиг. I).



Фиг. 1. Языки, значения, функции.

Определим теперь лаконичность ПОЯ как свойство ПОЯ не содержать лишних символов, т.е. символов, которые не способствуют уяснению значения слова (запроса) ПОЯ. Например, пусть $P_1 \subseteq \{\omega_1 \omega \mid \omega \in \Sigma^*\}$ для фиксированного $\omega_1 \in \Sigma^*$.

Ясно, что если ограничиваться только языком P_1, то при любом разумном определении функции δ_{P_1} цепочка ω_1, предшествующая всем словам P_1, не поможет нам распознавать значение слова из P_1. Таким образом, ее можно было бы отбросить. Значение того, что P_1 порожден языком L, проявляется двояко: (I) в словосочетании "можно было бы отбросить" (именно, цепочка ω_1 может отказаться существенной, если рассматривать ее в контексте языка L), и (2) в том, что буквы языка P заимствованы из алфавита Σ (таким образом, язык $P_2 = \{$ЛАКОНИЧНЫЙ, НЕЛАКОНИЧНЫЙ$\}$, вероятно, лаконичен; экономно ли такое представление - этот вопрос зависит от другого окружения чем язык L и в данном случае не рассматривается). Отметим еще, что если δ_{P_2} (ЛАКОНИЧНЫЙ) $= \delta_{P_2}$ (НЕЛАКОНИЧНЫЙ), то язык P_2 нелаконичен.

Чтобы понятие лаконичности можно было использовать практически, его следует определить как меру. При этом целесообразно исходить не из языка как такового, а из реальной частоты использования слов языка в рассматриваемом применении. Именно, пусть $P = \bigcup_{i=1}^{k} P_i$; минимальная (относительно функции δ_P) длина слова $x \in P_i$ и ее действительная длина равны A_i, B_i соответственно; вероятность появления запроса $x \in P_i$ равна F_i. Тогда лаконичность определяется как

$$\frac{\text{минимальная средняя длина запроса}}{\text{действительная средняя длина запроса}} = \frac{\Sigma A_i F_i}{\Sigma B_i F_i}.$$

Пример. Пусть P задается правилами

E :: = РАЙОН = ⟨номер⟩
⟨номер⟩ :: = ⟨цифра⟩ | ⟨номер⟩ ⟨цифра⟩
⟨цифра⟩ :: = 0|I|2|3|4|5|6|7|8|9

Пусть вероятность появления в данном применении одноместных номеров района равна $F_1 = 0.3$, двухместных – $F_2 = 0.7$; РАЙОН – лексема языка L. Тогда для одноместных номеров $A_1 = 1, B_1 = 3$; для двухместных $A_2 = 2, B_2 = 4$ и лаконичность равна

$$\frac{0.3 \times I + 0.7 \times 2}{0.3 \times 3 + 0.7 \times 4} = \frac{I.7}{3.7},$$

т.е. запрос в среднем более двух раз длиннее, чем необходимо.

Заметим, что мы могли бы оценить вероятность p_i появления i-го слова языка P, перекодировать слова P в код минимальной средней длины $L_{min} = -\Sigma p_i \log_2 p_i$ [8] и определить лаконичность как

$$\frac{L_{min}}{\text{средняя длина программы}}.$$

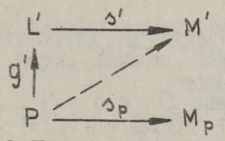
На первый взгляд кажется, что перекодирование в ПОЯ недопустимо (теряется проблемная ориентированность языка). Действительно же частое появление некоторого запроса показывает, что этот запрос представляет важную самостоятельную проблему, т.е. его как раз следовало бы выделить (под подходящим именем) как отдельный объект ПОЯ. Такое выделение может быть частью процесса образования ПОЯ.

Понятность - понятие относительное, зависящее от объема наших знаний о данной предметной области. Если мы не знаем, что такое "отчет", то запрос "составить отчет по форме КI80 в разрезе района Октябрьский" может быть также непонятным, как и запрос "К I80, октябрьский". Это значит, что дополнительная информация, включаемая в запрос, должна быть содержательной для возможно более широкого круга пользователей. В данной работе предполагается, что такая содержательность достигается за счет выбора подходящих конструкций базового языка L.

Итак, понятность ПОЯ зависит от уровня знаний пользователя, от его собственных представлений о L, M, g и δ - другими словами, от пользовательской модели предметной области и соответствующего программного обеспечения (далее коротко "пользовательская модель ПО").

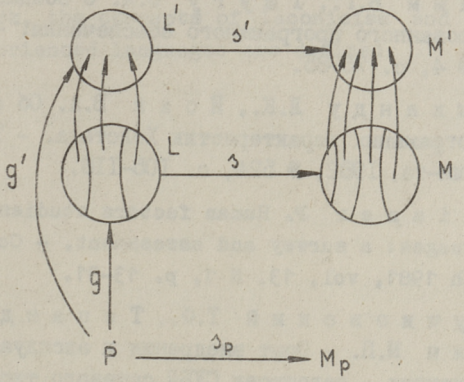
Интуитивно, программа на ПОЯ является понятной для данного пользователя, если эта программа, рассмотренная в контексте знаний пользователя, полностью определяет решаемую задачу, т.е. если пользовательская модель ПО включает функцию с языка P во множество значений M.

При каких условиях это возможно? Пользователь ПОЯ - обычно специалист предметной области. Это значит, что он знает свои задачи и их возможные решения. Если язык L создан правильно, то в нем отражается терминология предметной области - слово у языка L представляет собой некоторую задачу из предметной области. Соответственно, значение $\delta(y)$ из M - решение задачи y. Поэтому можно предположить, что модель пользователя включает L', δ', M' - более или менее адекватное представление L, δ, M. Кроме того, он должен знать язык P и функцию δ_p, так как он использует язык P. Наконец, у него должно быть некоторое представление о том, какие подмножества L' соответствуют словам P; обозначим соответствующую функцию через g' (фиг. 2). Отметим, что для лаконичных языков P отображение g' может каждому элементу из P поставить в соответствие множество L' (запрос непонятен), а для понятных языков с каждым элементом из P может сопоставляться один элемент (запрос полностью определяет проблему).



Фиг. 2. Пользовательская модель предметной области.

Мы можем делать еще следующие предположения о характере L', $\mathfrak{s}'$ и M' (фиг. 3): человек не может удерживать в памяти всевозможные слова и их значения. Скорее всего, L' и M' разумно представить как факторные множества некоторого разбиения множеств L и M на сходные между собой элементы. При этом элементы из L' и M' могут рассматриваться как общие имена совокупностей проблем и их возможных решений. Например, проблема "заболеваемость", сформулированная как "находится ли заболеваемость болезнью X в городе Y в допустимых пределах" - на самом деле группа проблем, зависящих от X и Y. Пользователь знает значение $\mathfrak{s}_P(x)$ запроса $x \in P$; ему также известен характер задачи $g'(x)$ и решения $\mathfrak{s}'(g'(x))$. Если этих знаний достаточно для получения ответа (это не значит, что он должен сам вычислять ответ), то запрос x ему понятен.



Фиг. 3. Соответствие программного обеспечения и ее пользовательской модели.

Таким образом, мы приходим к следующей формулировке: язык P называется понятным, если существует функция f, такая, что для каждого $x \in P$,

$$f(\sigma_p(x),\sigma'(g'(x)))=\sigma(g(x)).$$

Например, язык вида

ЗАБОЛЕВАЕМОСТЬ # ГОРОД = ... # БОЛЕЗНЬ = ...

понятен для человека, знающего проблему "ЗАБОЛЕВАЕМОСТЬ" (функции f, g', σ') и то, где найти информацию о заданном городе и болезни (функция σ_p).

Заключение. Предложенный способ генерации ПОЯ с общим ядром может использоваться в разных системах обработки данных. Критерии оценки ПОЯ могут быть полезными при спроектировании, реализации и оценке ПОЯ. При разработке пакетов программ, минимизирующих требования к объему знаний пользователя, необходимы (пользовательские) модели предметной области и программного обеспечения.

Литература

1. Kerninghan B.W., Plauger P.I. Software tools. Addison - Wesley Publishing Company, 1976. 338 p.

2. Тамм Б.Г., Тыугу Э.Х. О создании проблемно-ориентированного программного обеспечения. - Кибернетика, 1975, № 4, с. 76-85.

3. Выханду Л.К., Йокк В.А. Об опыте исследований программных характеристик Холстеда. - Тр. Таллинск. политехн. ин-та, 1982, № 524, с. 105-118.

4. Reisner P. Human factors studies of database query languages: a survey and assessment. - Computing surveys, March 1981, vol, 13, N 1, p. 13-31.

5. Лучковский Т.Ф., Тепанди Я.Я., Хермлин М.П. - Опыт внедрения и эксплуатации систем обработки данных - надстроек СУБД сетевого типа. - Тр. Таллинск. политехн. ин-та, 1983, № 554, с. 3-11.

6. Вольдман Г.Ш. Объяснение непроцедурности средствами теории категорий. - Программирование, 1980, № 4, с. 72-79.

7. Тепанди Я.Я. Исследование и разработка проблемно-ориентированных систем обработки данных, основанных

на системах управления базами данных сетевого типа. Дис. на соиск. канд. техн. наук. Таллин, 1982.

8. Бауэр Ф.Л., Гоос Г. Информатика. М., Мир, 1976. 484 с.

J. Tepandi

Assessment Criteria for Languages with a Common Kernel

Summary

A method for generating problem-oriented languages with a common kernel-language is presented. According to this method, some gaps are left in the kernel-language programs. The gaps are filled up before executing the program. The necessary text forms the problem-oriented language under review. A user's model of the computer system is given. In this framework, the concepts of laconicism and clearness of the problem-oriented languages are defined.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.518:519.682

У.А. Пауклин, Ю.Э. Эйвак

ВОЗМОЖНОСТИ ЗАПРОСА ДАННЫХ НА ЯЗЫКЕ DAMAL СИСТЕМЫ ПАРЕС

Язык манипулирования данными (ЯМД) DAMAL является одним из входных языков системы ПАРЕС, разработанной в Таллинском политехническом институте [1]. Эта система представляет собой инструментальную систему программирования, предназначенную для создания прикладных систем обработки информации с интегрированной базой данных (БД) реляционно-решетчатой структуры [2, 3].

ЯМД DAMAL является языком программирования высокого уровня, сочетающим основные функции базового языка, языка запросов и ЯМД общеизвестных систем управления базами данных (СУБД) [4].

В данной статье рассматриваются возможности языка DAMAL по поиску информации в базе. С этой целью после краткой характеристики модели данных указанной структуры и самого языка DAMAL приводится ряд примеров по его применению в качестве языка запросов. При этом в роли эталона сравнения используется язык ALPHA, предложенный Коддом для реляционной модели [5, с. 67-75].

Реляционно-решетчатая модель (РРМ) БД является, в некотором смысле, обобщением сетевой модели КОДАСИЛ и реляционной модели Кодда. По РРМ глобальная структура БД представляется ориентированным графом, вершины которого называются реляционными объектами (РО), а дуги - реляционными связями (РС). Предполагается, что этот граф является частично-упорядоченным, и что в нем определены наибольший и наименьший элементы, называемые корнем и атомом соответственно. Предполагается также, что за каждым РО скрывается целое множество однотипных элементов (объектов), за исключением корня

модели, который всегда представляется как одноэлементное множество; каждая дуга графа понимается как образ некоторого I:M-соответствия между объектами соответствующих множеств.

Помимо глобальной структуры БД говорят еще о ее локальной структуре (внутренней структуре РО). Элементами этой структуры являются атрибуты объектов.

Все РО в качестве элементов РРМ интерпретируются как образы записей БД соответствующего типа, атрибуты их - как поля данных в этих записях, а РС - как связи между записями соответствующих типов, реализуемые, например, той или другой системой ссылок. Такая интерпретация соответствует сетевой модели КОДАСИЛ. Но РРМ может легко интерпретироваться и как реляционная модель Кодда. При этом РО представляют отдельные отношения, а РС - пары ключевых атрибутов соответствующих отношений.

ЯМД DAMAL является алгоритмическим языком, основанным на идеях структурного программирования и исчисления предикатов. Он содержит средства трех основных видов:

1) составные операции для структурирования программ;
2) логические условия для поиска и отбора объектов (записей) БД;
3) простые операции для обмена и манипулирования данными.

Составные операции языка - конкатенация, итерация и альтернация - определяются по схемам (1)-(3) соответственно:

```
(1) имя: BEGIN                    (2) имя: WHILE  <условие>
        <процесс>                             DO <процесс>
         END имя;                             END имя;
(3) имя: IF <условие>
         THEN <процесс-I>
        [ELSE <процесс-I>]
         END имя;
```

где имя - идентификатор операции, <процесс> - последовательность операций, <условие> - некоторое логическое условие.

На основе итерации и альтернации строится еще ряд составных операций, которые, помимо функции структурирования программ, связаны с поиском объектов в БД. Например, операция поиска объекта по ключу идентификации - схема (4), операции последовательного поиска одного или нескольких объектов по реляционной связи - схемы (5) и (6) соответственно и др.:

```
(4) имя: IF EX ANY объект (ключ)
         [ SUCH <предикат> ]
           THEN <процесс-I>
         [ ELSE <процесс-2>]
           END имя;

(5) имя: IF EX (член IN глава)
         [SUCH <предикат>]
          THEN <процесс-I>
         [ELSE <процесс-2>]
          END имя;

(6) имя: FOR ALL (член IN глава)
           [SUCH <предикат>]
            DO <процесс>
            END имя;
```

где объект - имя объекта, ключ - ключ идентификации объекта, член и глава - имена члена и главы некоторой реляционной связи, <предикат> - условие поиска.

Логические условия DAMAL'а делятся на простые и составные. Составные логические условия представляются в виде формул исчисления предикатов и состоят из простых или семантически более простых составных условий, соединенных логическими союзами NOT,AND и OR. В их состав могут входить также кванторы существования и всеобщности, приложенные либо к членам (схемы (7) и (8)), либо к главе некоторой реляционной связи (схема (9)).

```
(7) EX  (член IN глава)  [<предикат>]
(8) ALL (член IN глава)  [<предикат>]
(9) EX  (глава OF член)  [<предикат>]
```

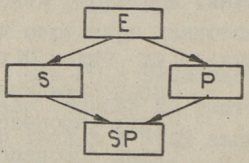
Простые логические условия могут быть двух видов - сравнение объектов БД (например, по схеме (I0)) и сравнение атрибутов объектов (что совпадает со сравнением данных в других языках программирования).

```
(I0) NEXT (член IN глава) = CUR (объект)
```

Простые операции обмена и манипулирования данными - к ним относится ряд стандартных для СУБД операторов манипули-

рования данными, таких, как операторы прибавления (INSERT), замены (REPLACE) и др., а также ряд стандартных операторов обработки данных - оператор присваивания (SET) операторы ввода-вывода (GET, PUT) и др.

Примеры запросов относятся к хорошо известной системе поставки изделий, описываемой моделью:



E = Ø
S (SNO, NAME, ST, CITY)⊂E
P (PNO, NAME, COL, WT, CITY)⊂E
SP (SNO, PNO, QTY)⊂S×P

где E - корневой объект, представляющий данную проблемную область; S - поставщик; P - изделие; SP - поставка; SNO и PNO - номера поставщика и изделия соответственно; NAME - имя; ST(STATUS) - статус; CITY - город; COL(COLOUR) - цвет; WT(WEIGHT) - вес; QTY(QUANTITY) - количество.

Приводимые примеры выбраны из набора запросов, составленного Дейтом в целях демонстрации возможностей языка ALPHA [5, с. 76-94]. Для предоставления возможности сравнения запросы оформлены на двух языках - на языке ALPHA и на языке DAMAL.

1) Получить (напечатать) все сведения о всех поставщиках.

```
A L P H A :
GET W (S)

D A M A L :
P: FOR ALL (S IN E)
       DO
         PUT (S.SNO, S.NAME, S.ST, S.CITY);
       END P;
```

2) получить номера поставщиков, находящихся в Париже и имеющих статус более 20.

```
A L P H A :
GET W (S.SNO) : S.CITY = 'PARIS'∧S.ST > 20

D A M A L :
P: FOR ALL (S IN E)
       SUCH S.CITY = 'PARIS' AND S.ST > 20
       DO
         PUT (S.SNO);
       END P;
```

3) Получить номер любого одного поставщика из числа поставщиков, находящихся в Париже.

```
A L P H A :
GET W(1)(S.SNO) : S.CITY = 'PARIS'

D A M A L :
P: IF EX (S IN E)
   SUCH S.CITY = 'PARIS'
   THEN
       PUT (S.SNO);
   ELSE
       PUT ('В БД НЕТ ПОСТАВЩИКА, НАХОДЯЩЕГОСЯ В ПАРИЖЕ');
   END P;
```

4) Получить номера всех поставляемых деталей.

```
A L P H A :
GET W (SP.PNO)
D A M A L :
P: FOR ALL (P IN E)
       SUCH EX (SP IN P)
       DO
         PUT (P.PNO);
       END P;
```

5) Получить имена тех поставщиков, которые поставляют деталь P2.

```
A L P H A :
RANGE SP X
GET W (S.NAME) : ∃X (X.SNO = S.SNO∧X.PNO = 'P2')

D A M A L :
P: FOR ALL (S IN E)
       SUCH EX (SP IN S)
            (SP.PNO = 'P2')
       DO
         PUT (S.NAME);
       END P;
```

6) Получить названия городов, в которых находятся поставщики, поставляющие деталь P2.

```
A L P H A :
RANGE SP Z
GET W(S.CITY) : ∃Z (Z.SNO = S.SNO∧Z.PNO = 'P2')
```

```
D A M A L :
P: FOR ALL (S IN E)
      SUCH EX (SP IN S)
           (SP.PNO = 'P2')
      DO
        PUT (S.CITY);
      END P;
```

В данном случае результаты запроса на языках ALPHA и DAMAL не в точности совпадают. Результатом на языке ALPHA является отношение в целом, в котором уже по определению не допускаются одинаковые строки. А на языке DAMAL результат запроса выдается по отдельным экземплярам PO, причем функции сжатия и представления результата выведены за пределы операции поиска. Поэтому, в последнем случае, некоторые названия городов могут появиться в выдаваемом списке несколько раз.

7) Получить номера тех поставщиков, которые не поставляют деталь PI.

```
A L P H A :
RANGE SP SPX
GET W (S.SNO) :∀SPX (SPX.SNO ¬= S.SNO∨SPX.PNO¬= 'P1')

D A M A L :
P: FOR ALL (S IN E)
      SUCH ALL (SP IN S)
           (SP.PNO ¬= 'P1')
      DO
        PUT (S.SNO);
      END P;
```

8) Получить номера поставщиков, которые поставляют хотя бы одну красную деталь.

```
A L P H A :
RANGE P PX
GET W (SP.SNO) :∃PX (PX.PNO = SP.PNO∧PX.COL = 'RED')

D A M A L :
P: FOR ALL (S IN E)
      SUCH EX (SP IN S)
           (EX (P OF SP)
             (P.COL = 'RED'))
```

```
        DO
          PUT (S.SNO);
        END P;
```

9) Получить имена тех поставщиков, которые поставляют хотя бы одну деталь, поставляемую поставщиком S2.

```
A L P H A :
RANGE SP SPX
RANGE SP SPY
GET W (S.NAME)  : ∃ SPX (SPX.SNO = S.SNO ∧
                    ∃ SPY (SPY.PNO = SPX.PNO ∧
                          SPY.SNO = 'S2'))

D A M A L :
P: FOR ALL (S IN E)
       SUCH EX (P IN E)
             (EX (SP IN P)(SP.SNO = 'S2')
              AND EX (SP IN P)(SP.SNO = S.SNO))
       DO
         PUT (S.NAME);
       END P;
```

10) Получить все такие пары номер поставщика/номер детали (PNO/SNO), в которых образующим их значением SNO и PNO соответствует одно и то же значение атрибута CITY.

```
A L P H A :
GET W (S.SNO, P.PNO) : S.CITY = P.CITY

D A M A L :
```

По заданию требуется формировать экземпляры нового, неопределенного в модели объекта. В подобных случаях на DAMAL'е потребуется более одной (в данном случае две) вложенных одна в другую операций:

```
P: FOR ALL (S IN E)
       DO
       P1: FOR ALL (P IN E)
              SUCH P.CITY = S.CITY
              DO
                PUT (P.PNO,S.SNO);
              END P1;
       END P;
```

11) Получить имена тех поставщиков, которые поставляют все детали.

```
A L P H A :
RANGE P PX
RANGE SP SPX
GET W (S.NAME): ∀PX ∃SPX (SPX.SNO = S.SNO ∧ SPX.PNO=PX.PNO)

D A M A L :
P: FOR ALL (S IN E)
       SUCH ALL (P IN E)
            (EX (SP IN P)
             (SP.SNO = S.SNO))
       DO
         PUT (S.NAME);
```

I2) Получить имена тех поставщиков, которые поставляют, по крайней мере, все те детали, что и поставщик S2.

```
A L P H A :
RANGE P PX
RANGE SP SPX
RANGE SP SPY
GET W (S.NAME): ∀PX(∃SPX (SPX.SNO = 'S2'∧SPX.PNO =PX.PNO)
                    →∃SPY (SPY.SNO = S.SNO∧
                           SPY.PNO = PX.PNO))

D A M A L :
P: FOR ALL (S IN E)
       SUCH ALL (P IN E)
         (ALL (SP IN P)(SP.SNO ¬= 'S2')
          OR EX (SP IN P)(SP.SNO = S.SNO))
       DO
         PUT (S.NAME);
       END P;
```

Заключение. Логические возможности обоих языков, ALPHA и DAMAL, по поиску данных из базы одинаковы. Оба языка оперируют одинаковыми, соответствующими друг другу операндами - кортежами отношений или экземплярами PO соответственно. Но выдаваемый результат (ответ на запрос) определен в этих языках по разному - в случае ALPHA как некоторое новое отношение, в случае DAMAL как последовательность строк данных, образованная на базе отфильтрованных экземпляров PO. Для сжатия и представления данных, а также для переупорядочения, подсчета и т.п. на DAMAL'е предусмотрены другие средства и возможности.

В приведенном анализе было рассмотрено только одно свойство языка DAMAL - его селективная мощность, что является совершенно недостаточным для оценки возможностей, достоинств и недостатков языка в целом. Поскольку ЯМД DAMAL является одновременно и базовым языком, то его следовало бы сравнить не только с ALPHA в отдельности, а в сочетании с каким-либо базовым языком, например, с PL или COBOL. Но подобное сравнение выходит за рамки данной работы. Более подробно язык DAMAL описан в [4].

Литература

I. Крахт В.А., Эйвак Ю.Э. Система ведения баз данных и манипулирования данными "ПАРЕС". - Таллин,ТПИ, 1979. 37 с.

2. Крахт В.А. Реляционно-решетчатая модель для представления концептуальной структуры предметной области АСУ в базах данных. - Сб. Кибернетика и вуз. / Томский политехн. ин-т - Томск: ТПИ, 1980, вып. 15, с. 134-144.

3. Крахт В.А., Роотталу Э.П. Проектирование баз данных на основе реляционно-решетчатой концептуальной модели предметной области. - УСиМ, 1981, № 4, с. 22-28.

4. Крахт В.А., Эйвак Ю.Э. Алгоритмический язык манипулирования данными DAMAL. - Таллин, ТПИ, 1982. 118 с.

5. Дейт К. Введение в системы баз данных. М., Наука, 1980. 463 с.

U. Pauklin, J. Eivak

The Possibilities of Data Query in PARES System Data Manipulation Language DAMAL

Summary

The range of DAMAL usage as a query language is discussed. Data model and the language are described and some examples are presented. ALPHA is used as a standard for comparison.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.016

А.В. Рензер

СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ЭКОНОМИЧЕСКОГО АНАЛИЗА НА БАЗЕ ПОК "АРМ-ЭКОНОМИКА"

I. Введение

Современное управление предприятием и научно обоснованное планирование работы требуют глубокого анализа экономической деятельности предприятия. Средством автоматизации этого процесса являются ЭВМ разных мощностей в зависимости от уровня и тщательности проведения экономического анализа в различных звеньях управления [см. I].

Основной узел системы автоматизированного экономического анализа - это автоматизированное рабочее место (АРМ) различных специалистов и руководителей, которое обеспечивает их информацией как о текущем ходе экономической деятельности, так и об общем состоянии предприятия. Каждое АРМ может иметь специфические возможности и ограничения в зависимости от его назначения и уровня защиты информации, но тем не менее все АРМ должны исходить из одной и той же информационной базы.

До настоящего времени все системы организационно-экономического характера выполнялись как индивидуальные разработки для конкретного предприятия на базе средних и больших ЭВМ. Отсутствовала концепция полной машинной обработки и хранения информации, всегда предполагался специальный этап подготовки и ввода данных с первичных документов [I, 3]. Наступила пора разработки систем, которые предусматривают первичную обработку данных (ввод, исправление, рассортировка) исключительно на мини- и микро-ЭВМ с последующей обработкой (статистические отчеты и тиражирование) на средних и больших ЭВМ в зависимости от нужд конкретной организации. Такие системы должны быть универсальными, т.е. они должны обеспечить обработку данных любых организаций, а составляемые этими систе-

мами отчеты должны удовлетворять всем требованиям, которые предъявляются к официальным документам.

Разрабатываемая в ТПИ первая часть универсальной системы автоматизированного экономического анализа базируется на вычислительных машинах типа СМ-1800. Основой математического обеспечения системы является проблемно-ориентированный комплекс (ПОК) "АРМ-экономика", который дает наиболее широкие возможности для организации диалога с подсистемами и для обработки различных документов.

2. Базисная система ПОК "АРМ-экономика"

Поскольку ПОК "АРМ-экономика" в достаточной мере описывается в книге [3], в данной статье приводятся только самые необходимые сведения об этой системе.

ПОК "АРМ-экономика" состоит из технических средств (для нас - это СМ-1803, архитектура которой в статье не рассматривается), и специального системного программного обеспечения (СПО) для этих технических средств.

СПО ПОК "АРМ-экономика" состоит из трех частей:
- модули ядра системы,
- системные обслуживающие программы,
- комплекс проблемно-ориентированных программ.

Ядро системы управляет всем вычислительным комплексом, распределяет ресурсы, предоставляет возможности ведения диалога, ведения учета работ, выполняет защиту данных, а также содержит средства управления базой данных. Ядро загружается с дискетты и хранится в оперативной памяти на всем протяжении работы ПОК.

Системные обслуживающие программы предназначены для манипулирования данными на дискеттах (обслуживание файлов), и для разработки и сборки дополнительных программ на основе специального расширенного ассемблера и редактора связей.

Вся информация, введенная в систему пользователем, хранится в библиотечных файлах. Обрабатываемые документы записываются в банк данных, который также организован на библиотечных файлах.

Работа с ПОК "АРМ-экономика" не требует от пользователя глубоких знаний в области программирования. После запуска системы (загрузка ядра, сообщение текущей даты и времени, объявление пользователя) система находится в т.н. системном режиме, в котором всегда можно вызвать меню разрешенных действий (например, манипулирование дискеттами и файлами, включение и выключение печати, выполнение какой-то программы). Кстати, ведение диалога с помощью ядра системы построено так, что на любой системный запрос выбора ответа можно вызвать меню разрешенных ответов. Этот принцип использован и в системе автоматизированного экономического анализа.

Кроме процедур ядра, программ обслуживания дискетт и файлов, средств разработки новых программ для построения системы экономического анализа использованы и некоторые модули проблемно-ориентированных программ базисной системы.

Основная программа обработки табличных документов обеспечивает создание, заполнение данными, исправление и распечатку различных табличных документов. Предусмотрена и возможность формирования новых документов на основе имеющихся в базе экземпляров.

Описание табличного документа составляется заранее с помощью редактора текстовой информации и транслируется специальным транслятором описаний документов (ОД). Не вникая в подробности ОД отметим, что описания создаются на основе системы справочников, которые можно изменять независимо от того, объявлялись ли они уже в ОД какого-то документа или нет. Таким образом, после создания типового документа истинные данные в заголовке, шапке и боковине документа могут быть определены пользователем самостоятельно. Составление же ОД требует более глубоких знаний о системе и,по всей видимости, останется на долю администратора системы [см. I].

Некоторые значения полей в документе можно не вводить, а вычислять по правилам, которые подсказывает пользователь. Для этого, а также для выполнения логического контроля над вводимыми данными, в ОД включаются различные формулы, которые описывают взаимосвязь между некоторыми полями документа. Формулы представляют на специальном языке, овладение которым требует также определенных знаний о системе описывания

и хранения документов. Поэтому надо полагать, что и эту работу должен выполнять администратор системы. Возможность изменения справочников в произвольное время упрощает коренным образом настройку системы экономического анализа на конкретную организацию.

Другая проблемно-ориентированная программа базисной системы, которая находит применение в системе экономического анализа - это т.н. малая информационно-справочная система. С помощью последней можно на естественном языке хранить, например, полные описания экономических показателей и точные формулы их вычисления. В случае, если пользователь во время работы сталкивается с трудностями, он может легко получить вспомогательную информацию с помощью этой системы.

3. Архитектура системы автоматизированного экономического анализа

Основные концепции подобных систем изложены уже в статьях [1] и [2]. Различаются три уровня работы по анализу экономической деятельности организации:

- повседневная работа с документами показателей;
- случайные запросы различных одиночных показателей;
- выполнение специальных расчетов.

Документы показателей составляются логическими целыми по принципу соединения между собой зависимых друг от друга показателей. Обычно составляется один документ на один раздел комплексного экономического анализа [см. 4]. В одном документе используются показатели двух типов - исходные и вычисляемые. Значения исходных показателей вводятся в документ оператором прямо на дисплее, значения вычисляемых показателей находит сама система по исходным показателям. Правила вычисления определены в описании документа специальными формулами.

Все данные анализа, т.е. показатели, хранятся в базе данных, где любой показатель определен документом, в котором он описан, и собственным кодом внутри этого документа. При запросе одиночного показателя необходимо установить код показателя и название соответствующего документа.

Специальные расчеты выполняются по различным алгоритмам проведения экономико-статистических расчетов.

Подробная классификация пользователей информационных систем давалась уже в статье [2]. Напомним, что администратором системы является специальный пользователь, задача которого состоит в настройке, обслуживании и сопровождении системы. Он должен в равной мере быть специалистом как в сфере обработки информации, так и в экономической деятельности конкретного предприятия. Конечный пользователь не обязан иметь квалификацию программиста, он общается с системой на естественном языке. Именно он нуждается в информации, которая вводится, хранится и обрабатывается с помощью автоматизированной системы. Как уже отмечалось, разные конечные пользователи работают с разными подмножествами данных и поэтому могут иметь разные взгляды на общую схему хранимой информации, но все АРМ должны исходить из одной информационной базы. Необходимые преобразования данных не должны быть видны конечному пользователю.

Система автоматизированного экономического анализа организована в виде рабочей программы "АРМ-экономики" и загружается системными средствами базисной системы.

Сеанс начинается с вывода на дисплей сообщения о начале работы системы экономического анализа. Затем у пользователя спрашивается режим работы. Напомним, что на любой вопрос системы можно вызвать меню возможных ответов, который в данном случае был бы следующим:

- (минус) - конец работы системы экономического анализа,
Д - работа с документами показателей,
П - работа с одиночными показателями,
Р - работа со специальными программами анализа,
Х - вспомогательная информация.

При завершении работы системы экономического анализа производится выход в системный режим "АРМ-экономики", и процесс остается в режиме ожидания команд.

При выборе режима вспомогательной информации активизируется малая информационно-справочная система.

Специальные программы экономического анализа включают несколько специфических модулей общего назначения для вы-

полнения расчетов по наиболее распространенным алгоритмам. Например, имеются программы для матричного анализа по моделям проф. У. Мересте [см. 5].

При выборе режима работы с документами показателей система запрашивает вид работы. Возможные ответы:

- (минус) - выход на выбор режима,
К - каталог имеющихся документов,
В - ввод документа и нахождение вычисляемых показателей,
И - исправление с перевычислением показателей,
П - просмотр документа.

Просмотр документа может осуществляться либо с распечаткой на мозаичном печатающем устройстве, либо без этого. Система запрашивает ответ на соответствующий вопрос.

При вводе, исправлении и просмотре документа дополнительно спрашивается название документа, к этому времени выводится на экран синтаксически правильный трафарет. Если название документа неверное (или при режимах просмотра или исправления такого документа в базе не оказывается), система спрашивает, повторить ли ввод названия, отказаться ли от этого вида работы или выдавать вспомогательную информацию (каталог имеющихся документов, справочники организаций и годов).

После выяснения названия документа активизируются модули, соответствующие выбранному виду работы. При вводе нового документа бланкет соответствующего типа документов (исходные показатели) заполняется на экране и после этого вычисляются значения вычисляемых показателей. Готовый документ вводится в базу данных, при обнаружении ошибок его можно либо исправлять, либо уничтожать с помощью специальной программы.

При выборе режима работы с одиночными показателями у пользователя последовательно спрашиваются раздел анализа (обычно это соответствует одному типу документов), название организации и код года. Таким образом, по существу определяется один документ показателей, в котором будут искать показатели по кодам. Любой показатель идентифицирует-

ся типом (I-исходный, 2-вычисляемый) и кодом из справочника показателей такого типа. Значение искомого показателя выводится на экран и после этого система спрашивает, как продолжить работу. Возможные ответы:

- (минус) - выход из режима
П - иной показатель
Г - иной год
О - иная организация

При ответе П, Г или О система потребует новые соответствующие названия или коды, и работа продолжится.

4. Пример использования системы автоматизированного экономического анализа

Покажем использование системы для некоторого абстрактного предприятия, которое состоит из двух цехов и где анализируется только использование рабочей силы. В таком случае необходимо описать только один тип документов показателей - назовем этот тип РАСИ.

Нужны справочники цехов, годов (периодов), показателей обоих типов. Справочник состоит из множества рядов, где каждый ряд состоит из кода и соответственного текста - см. [3]:

ЦЕХА:	I	ЦЕХ-I
	2	ЦЕХ-2
ГОДЫ:	82	I982.Г.
	83	I983.Г.
	84	I984.Г.
ИСХПОК:	0I0	РАБОЧИХ В БОЛЬШЕЙ СМЕНЕ
	020	РАБОЧИХ ДНЕЙ В ПЕРИОДЕ
	030	ЧЕЛОВЕКОДНЕЙ
	040	ЧИСЛЕННОСТЬ РАБОЧИХ
	050	ЧИСЛЕННОСТЬ РАБОТНИКОВ
	060	СРЕДНИЙ ТАРИФНЫЙ РАЗРЯД РАБОЧИХ
	065	СРЕДНИЙ ТАРИФНЫЙ РАЗРЯД РАБОТ
	070	ПРИНЯТЫХ НА РАБОТУ
	080	УВОЛЕННЫХ
	090	УШЕДШИХ ПО СОБСТВЕННОМУ ЖЕЛАНИЮ
	I00	УШЕДШИХ ПО ИНЫМ ПРИЧИНАМ

```
ВЫЧПОК: 010  КОЭФФИЦИЕНТ СТАБИЛЬНОСТИ Р.С.
        020  КОЭФФИЦИЕНТ ИСПОЛЬЗОВАНИЯ СМЕНЫ
        030  КОЛ-ВО РАБОЧИХ, КВАЛИФИКАЦИЮ КОТОРЫХ
             НЕОБХОДИМО ПОВЫСИТЬ
        060  КОЭФФИЦИЕНТ ТЕКУЧЕСТИ Р.С.
        070  КОЭФФИЦИЕНТ ОБНОВЛЕНИЯ Р.С.
        080  КОЭФФИЦИЕНТ УХОДА Р.С.
        090  КОЭФФИЦИЕНТ ИСПОЛЬЗОВАНИЯ СРЕДНЕЙ
             ПОСПИСКОВОЙ ЧИСЛЕННОСТИ
```

В шапке документа 2 столбца - любой показатель может иметь плановое и/или фактическое значение. Нужен справочник

```
ПЛФТ:    1  ПЛАН
         2  ФАКТ
```

Используя еще промежуточный справочник ФИКТ и тексты из справочника ПОК, можно достичь двухуровневой боковины документа:

```
ФИКТ:    1  ИСХПОК
         2  ВЫЧПОК

ПОК:     1  ИСХОДНЫЕ ПОКАЗАТЕЛИ
         2  ВЫЧИСЛЯЕМЫЕ ПОКАЗАТЕЛИ
```

Описание типа документов РАСИ составляется из этих справочников. Приведем описание с комментариями.

```
; ОПРЕДЕЛЕНИЕ СПРАВОЧНИКОВ ЗАГОЛОВКА
СПЗ ЦЕХА,ГОДЫ
; ОПРЕДЕЛЕНИЕ СПРАВОЧНИКОВ БОКОВИНЫ
СПБ ПОК,ФИКТ(3)
; ОПРЕДЕЛЕНИЕ СПРАВОЧНИКОВ ШАПКИ
СПШ ПЛФТ
; СЛЕДУЮТ ФОРМУЛЫ ВЫЧИСЛЕНИЯ
Ф 2,010,2='I'- (I,090,2+I,080,2+I,100,2)/(I,050,2+I,070,2)
Ф 2,020,2=I,050,2/I,010,2
Ф 2,030,2=(I,065,2-I,060,2)*I,040,2
Ф 2,060,2=(I,090,2+I,080,2)/I,050,2
Ф 2,070,2=I,070,2/I,050,2
Ф 2,080,2=(I,090,2+I,080,2+I,100,2)/I,050,2
Ф 2,090,2=(I,030,2/I,020,2)-I,050,2
; ЗАПРЕЩЕНИЕ ПЕЧАТИ НОМЕРА ЛИСТА, ВРЕМЕНИ И ДАТЫ
БЕЗНЛ
```

```
БЕЗДАТ
БЕЗВРМ
; ТЕКСТ ПОСЛЕ ПЕЧАТИ ЗАГОЛОВКА
ПТПЗ 'АНАЛИЗ РАБОЧЕЙ СИЛЫ'
```

Описание документа транслируется, и можно начинать работать с системой экономического анализа. Одним документом показателей является документ на один цех за год (периодом может быть, конечно, и более короткий срок). Если потребуется информация обо всем предприятии или по нескольким годам (периодам) одновременно, можно либо сформировать такие документы с помощью основной программы обработки табличных документов, либо соединить документы временно операторами управления печатью [3].

Допустим, что созданы документы за 1982, 1983 годы на первый цех и за 1983, 1984 годы на второй.

Предлагаем протокол показательного сеанса. Ответы пользователя подчеркнуты.

```
..... СИСТЕМА ЭКОНОМИЧЕСКОГО АНАЛИЗА.....
РЕЖИМ?ДОКУМЕНТ ПОКАЗАТЕЛЕЙ
ВИД РАБОТЫ?КАТАЛОГ ИМЕЮЩИХСЯ ДОКУМЕНТОВ

КАТАЛОГ БИБЛИОТЕКИ ЭА/ДАННЫЕ                 11:09:52 08.05.83
NN СОЗДАН   ИЗМЕНЕН ТИП ИМЯ
0  09.03.83 09.03.83 Т
1  05.05.83 05.05.83 Т РАСИ,ЦЕХ-1,83
2  05,05.83 05.05.83 Т РАСИ,ЦЕХ-1,82
3  05.05.83 05.05.83 Т РАСИ,ЦЕХ-2,83
4  05.05.83 05.05.83 Т РАСИ,ЦЕХ-2,84
--- КОНЕЦ КАТАЛОГА ---

РЕЖИМ?ДОКУМЕНТ ПОКАЗАТЕЛЕЙ
ВИД РАБОТЫ?ПРОСМОТР ДОКУМЕНТА
ВЫДАВАТЬ НА МПУ??
Д: ДА
Н: НЕТ
-: ВЫХОД
ВЫДАВАТЬ НА МПУ?НЕТ
ВВЕДИТЕ НАЗВАНИЕ ДОКУМЕНТА
<РАЗДЕЛ-АНАЛИЗА,ОРГАНИЗАЦИЯ,ГОД>:
РАСИ,ЦЕХ-1,83
```

*** ОБРАБОТКА ДОКУМЕНТОВ *** 08.05.83 II:II:I2
08.05.83 II:II:I3 ВИД РАБОТЫ: ПЕЧАТЬ

ЦЕХ-I
I983.Г

АНАЛИЗ РАБОЧЕЙ СИЛЫ

	: ПЛАН :	ФАКТ
А	: I :	2
ИСХОДНЫЕ ПОКАЗАТЕЛИ:		
РАБОЧИХ В БОЛЬШЕЙ СМЕНЕ	470.0	46I.0
РАБОЧИХ ДНЕЙ В ПЕРИОДЕ	I45.0	I50.0
ЧЕЛОВЕКОДНЕЙ	I30000.0	II0000.0
ЧИСЛЕННОСТЬ РАБОТАЮЩИХ	623.0	6II.0
СРЕДНИЙ ТАРИФНЫЙ РАЗРЯД РАБОЧИХ		4,4
СРЕДНИЙ ТАРИФНЫЙ РАЗРЯД РАБОТ		4.3
ПРИНЯТЫХ НА РАБОТУ		I02.0
УВОЛЕННЫХ		II7.0
УШЕДШИХ ПО СОБСТВЕННОМУ ЖЕЛАНИЮ		5.0
УШЕДШИХ ПО ИНЫМ ПРИЧИНАМ		3.0
ВЫЧИСЛЯЕМЫЕ ПОКАЗАТЕЛИ:		
КОЭФФИЦИЕНТ СТАБИЛЬНОСТИ Р.С.		0.8
КОЭФФИЦИЕНТ ИСПОЛЬЗОВАНИЯ СМЕНЫ		I.4
КОЛ-ВО РАБОЧИХ, КВАЛИФИКАЦИЮ КОТОРЫХ НЕОБХОДИМО ПОВЫСИТЬ		-47.5
КОЭФФИЦИЕНТ ТЕКУЧЕСТИ Р.С.		0.2
КОЭФФИЦИЕНТ ОБНОВЛЕНИЯ Р.С.		0.2
КОЭФФИЦИЕНТ УХОДА Р.С.		0.2
КОЭФФИЦИЕНТ ИСПОЛЬЗОВАНИЯ СРЕДНЕЙ ПОИСКОВОЙ ЧИСЛЕННОСТИ		I22.3

ВИД РАБОТЫ?_ВЫХОД НА ВЫБОР РЕЖИМА
РЕЖИМ?ПРАБОТА С ОДИНОЧНЫМИ ПОКАЗАТЕЛЯМИ
РАЗДЕЛ АНАЛИЗА?РАБОЧАЯ СИЛА
ОРГАНИЗАЦИЯ?ЦЕХ-2
ГОД?84
ТИП И КОД ПОКАЗАТЕЛЯ <ТИП,КОД> : 2,010

ЦЕХ-2 I984.Г.	ПЛАН	ФАКТ
КОЭФФИЦИЕНТ СТАБИЛЬНОСТИ Р.С		0.7

КАК ПРОДОЛЖАТЬ?_ВЫХОД ИЗ РЕЖИМА
РЕЖИМ?_КОНЕЦ РАБОТЫ СИСТЕМЫ ЭКОНОМИЧЕСКОГО АНАЛИЗА

Литература

I. Рензер А.В. Система автоматизированного анализа экономической деятельности предприятия. - Тр. Таллинск. политехн. ин-та, 1983, № 554, с. 121-127.

2. Лучковский Т.Ф., Микли Т.И., Рензер А.В. Экономические информационные системы коллективного пользования. - Тр. Таллинск. политехн, ин-та, 1982, № 524, с. 23-27.

3. Система обработки экономической информации на малых ЭВМ / Под ред. Д.В. Юрина. М., Книга, 1981. 184 с.

4. Шеремет А.Д., Сайфулин Р.С. Методика комплексного анализа хозяйственной деятельности промышленного предприятия (объединения). М., Экономика. 1980. 232 с.

5. Mereste U. Ühiskondliku tootmise majanduslik efektiivsuse tõus Eestis 1960-1977. ENSV TA Toimetised, 1980, 29. kd., nr. 1.

A. Renzer

An Automatic Economical Activity Analysis System Based on the Problem-Oriented Complex "Economics"

Summary

In this article the main properties of an automatic economical analysis system are presented and the underlying complex "Economics" is reviewed. The analysis system is highly user-oriented and designed for a wide range of non-programmers.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.03

Х.Х. Рохтла

ОБ ОДНОМ МЕТОДЕ ОПРЕДЕЛЕНИЯ СХОЖЕСТИ ЦЕПОЧЕК СИМВОЛОВ

I. Введение

В [1] была предложена идея создания каскада средств обработки синтаксических ошибок (СО). Первым двум уровням каскада посвящены работы [2, 3, 4]. Но в статье [4], где был изложен метод локальной нейтрализации СО, одна проблема осталась открытой. На последнем этапе нейтрализации каждой ошибки некоторые части ошибочной фразы (находящиеся в виде цепочек символов языка в магазине анализатора) приходится сравнивать с правыми частями правил подстановки языка (являющимися также цепочками символов языка). Среди сравниваемых цепочек отыскиваются наиболее совпадающие. Критерием сравнения должно служить некоторое понятие меры схожести или различия цепочек символов. Нужно иметь в виду и то, что такую меру приходится вычислять десятки или даже сотни раз при обработке каждой СО. Таким образом, быстродействие становится решающим требованием к применяемому алгоритму. Кроме того нужно учесть то, что задачей является не точное определение схожести всех цепочек символов, а только выбор пары цепочек с наибольшей схожестью.

Следующий раздел данной статьи посвящается разным методам определения схожести символьных строк. В третьем разделе рассматривается понятие локальной меры различия цепочек символов и указывается, что она соответствует поставленным требованиям. В четвертом разделе дается алгоритм вычисления локальной меры различия, а в пятом - примеры работы алгоритма.

2. Методы определения схожести цепочек символов

Задача определения схожести слов появляется обычно при исправлении орфографических ошибок. Сравниваемыми цепочками являются в этом случае искаженное слово и слово из множества ключевых слов и/или идентификаторов. В большинстве работ для сравнения слов определяются различные понятия расстояния между словами.

По методу Моргана [6] действительное расстояние между словами $\mathfrak{X}$ и $\mathfrak{Y}$ - $D(\mathfrak{X},\mathfrak{Y})$ - не вычисляется. Проверяется только равенство $D(\mathfrak{X},\mathfrak{Y})=D_1$, где D_1 - расстояние между словами, имеющими разницу только в один символ. Другими словами, проверяется, можно ли получить слово $\mathfrak{X}$ от $\mathfrak{Y}$ с помощью пропуска, вставки, замены одного символа или перестановкой двух соседних символов. Метод отличается своим быстродействием, но не применим для вычисления расстояния, превосходящего D_1.

Идеи Моргана были развиты в работах Вагнера [7, 8]. Представлены алгоритмы для общего случая - определения расстояния между словами, имеющими неограниченное число несовпадающих символов. Аналогичные рассуждения приведены и в работах [9, 10, 11]. В последних двух статьях используется понятие взвешенного расстояния Левенштейна (WLD), определяемое следующим образом:

Пусть $\mathfrak{X}=a_1a_2\dots a_n$ и $\mathfrak{Y}=b_1b_2\dots b_m$ - две цепочки символов. Допустим, что $\mathfrak{X}$ можно привести к $\mathfrak{Y}$ заменой k_i символов, вставкой m_i символов и пропуском n_i символов. Тогда

$$WLD(\mathfrak{X},\mathfrak{Y})=\min_i(pk_i+qm_i+rn_i),$$

где p, q, r - неотрицательные весы соответственно для замены, вставки и пропуска символов.

$WLD(\mathfrak{X},\mathfrak{Y})$ вычисляется с помощью рекуррентных формул:

$$D_{i,j}=\min\left(D_{i-1,j}+q,\ D_{i-1,j-1}+p',\ D_{i,j-1}+r\right)$$
$$i\le m,\ j\le n,$$

где
$$m=|\mathfrak{Y}|+1,\quad n=|\mathfrak{X}|+1,$$

$$r'=\begin{cases}0, & \text{если } a_j=b_i\\ p, & \text{если } a_j\neq b_i\end{cases}$$

$$D_{i,1} = (i-1)\,q,$$
$$D_{1,j} = (j-1)\,r,$$
$$WLD(\mathfrak{X}, \mathfrak{Y}) = D_{m,n}.$$

Известно также расстояние Хэмминга - это частный случай расстояния Левенштейна, где учитывается только замена символов и $p = I$.

В работе Танака и Фу [II] расстояние Левенштейна модифицировалось - учитывались различные весы пропуска, вставки и замены каждого конкретного символа.

Как видно, все названные понятия расстояния определены как минимальные суммы цен некоторых элементарных исправлений, преобразующих одно слово в другое. Вычисление такого рода расстояний является задачей динамического программирования. Оно занимает время, пропорциональное произведению длин рассматриваемых слов и поэтому в наших целях не пригодно. Все же только такой подход дает действительно минимальное расстояние между словами. В следующем под минимальным расстоянием между $\mathfrak{X}$ и $\mathfrak{Y}$ понимается именно $WLD(\mathfrak{X}, \mathfrak{Y})$.

В работах Нееба и Сидорова [I2, 5] предлагается иной подход - сравнение схожести цепочек символов с помощью коэффициентов соответственно буквенной и списковой корреляции. Но и эти методы не удовлетворяют нашим требованиям к быстродействию.

В следующем разделе предлагается новое понятие для определения схожести цепочек символов - локальная мера их различия. По законам метрики это понятие нельзя называть расстоянием, хотя и оно вычисляется как сумма цен элементарных исправлений трех названных типов. Как будет видно, это вычисление занимает время $O(M+N)$, где M и N - длины рассматриваемых цепочек символов.

3. Определение локальной меры различия цепочек символов

Сформулируем точнее приведенные во введении требования к применяемому методу определения схожести цепочек символов:

I) линейное быстродействие;

2) возможность нахождения пары цепочек с наибольшей среди всех пар схожестью.

Как было видно в предыдущем разделе, вычисление минимального расстояния требует сравнения обеих цепочек в целом и из-за этого занимает время $O(N \cdot M)$. Можно выдвинуть гипотезу, что ускорения вычисления можно добиться сравнением между собой лишь локальных участков цепочек символов, хотя в этом случае может потеряться минимальность результата. Понятие локальной меры различия (ЛМР) цепочек символов основывается именно на этой идее.

Объясним вычисление ЛМР неформально. Из сравниваемых цепочек символов выделяют первую (искаженную) и называют исправляемым словом, а вторую (словарную) - исправляющим словом. Каждый символ исправляемого слова разыскивается отдельно в исправляющем слове, причем в последнем просматривается не более двух символов - текущий и при необходимости следующий за ним символ. Точнее начальная часть алгоритма работает следующим образом:

Пусть $\mathcal{Y} = y_1 y_2 \cdots y_m$ - исправляемое слово и
$\mathcal{X} = x_1 x_2 \cdots y_n$ - исправляющее слово,
начальное значение ЛМР $(\mathcal{Y}, \mathcal{X}) = 0$.

I. Символ y_1 сравнивается с символом x_1. Если символы совпали, переходят к просмотру символов y_2 и x_2 (шаг I).

2. Если y_1 и x_1 не совпали, сравниваются y_1 и x_2. Если они совпали, ЛМР $(\mathcal{Y}, \mathcal{X})$ увеличивается на цену вставки символа x_1 и переходят к просмотру символов y_2 и x_3 (шаг I).

3. Если y_1 и x_2 не совпали, переходят к сравнению символа y_2 с x_1 и x_2, имея в виду, что y_1 остался ненайденным (шаг 4).

4. Если y_2 и x_1 совпали, ЛМР $(\mathcal{Y}, \mathcal{X})$ увеличивается на цену пропуска символа y_1 и переходят к просмотру символов y_3 и x_2 (шаг I).

5. Если y_2 и x_2 совпали, ЛМР $(\mathcal{Y}, \mathcal{X})$ увеличивается на цену замены символа y_1 на x_1 и переходят к просмотру символов y_3 и x_3 (шаг I).

6. Если и y_2 и x_2 не совпали, ЛМР $(\mathcal{Y}, \mathcal{X})$ увеличивается на цену замены символа y_1 на x_1 и переходят к просмотру символов y_3 и x_2, имея в виду, что y_2 остался ненайденным (шаг 4).

Если при работе алгоритма встречаются концы исправляемого или исправляющего слова, то вычисленное до сих пор ЛМР увеличивается на цену соответственно вставки непросмотренных символов исправляющего слова или пропуска непросмотренных символов исправляемого слова.

Пошаговый локальный просмотр сравниваемых цепочек обеспечивает время работы алгоритма - $O(N+M)$, где N и M - длины рассматриваемых цепочек. Но это преимущество, как и ожидалось, оплачивается потерями в точности вычисления - результат не всегда равен минимальному расстоянию между теми же цепочками символов.

Допустим, что цены пропуска, вставки и замены каждого символа равны единице. Приведем некоторые пары слов и минимальные расстояния между этими словами:

СЛОВО - ЛОВО I
СЛОВО - СЛОВА I
СЛОВО - ЛОВОС 2
СЛОВО - СЛОН 2
СЛОВО - СЛИВКИ 3

Точно такие значения получаются при вычислении ЛМР этих слов. Но, например, при паре:

СОСНА - ОСНА I

ЛМР будет равной 5. Анализ показывает, что при возрастании минимального расстояния между словами возрастает и вероятность, что значение ЛМР отличается от минимального расстояния. Возникает вопрос, можно ли каким-либо способом предотвратить такие большие отклонения. Просмотр в исправляющем слове более двух символов не имеет смысла, так как отклонение результата может при этом даже увеличиваться. Некоторое "улучшение" значения ЛМР можно получить путем повторения вычисления в обратном направлении - начиная с концов просматриваемых слов. Из двух результатов вычисления выбирается минимальный. При этом вдвое увеличиваются затраты времени, но при минимальном расстоянии, равном единице, гарантируется такое же значение ЛМР. (При паре СОСНА - ОСНА этот прием дает ЛМР = I). Также "улучшаются" результаты вычисления более больших значений ЛМР.

В заключение рассмотрим соответствие ЛМР требованиям, приведенным в начале раздела.

Требование I: удовлетворено, так как время вычисления ЛМР - O(N+M).

Требование 2: удовлетворено при повторении вычисления в обратном направлении, если найдется пара с минимальным расстоянием, равным единице. В остальных случаях выбранная пара может не иметь наибольшую схожесть. Все же практика показывает, что вероятность такого отклонения невелика, а также, что при нейтрализации 70-80 % из всех синтаксических ошибок можно найти пары цепочек символов с ЛМР = I.

Следовательно, локальную меру различия цепочек символов можно считать пригодным понятием для использования при методе локальной нейтрализации СО. В следующем разделе дается точное определение понятия ЛМР в виде алгоритма его вычисления.

4. Алгоритм ЛМР

В алгоритме предусмотрено вычисление ЛМР цепочек символов только в прямом направлении. Для повторения работы алгоритма в обратном направлении необходимо в сравниваемых словах переставить символы в обратном порядке, или же обеспечить изменение значений индексов, указывающих не просматриваемые символы, в обратном порядке. Из двух результатов работы алгоритма выбирается наименьший.

Отметим, что учитывая наши требования можно еще уменьшить время работы представленного алгоритма:

I. Если вычисляемая ЛМР уже превысила установленный предел (или наименьшую из найденных до сих пор ЛМР), то вычисление можно прерывать.

2. До начала алгоритма можно сравнить длины рассматриваемых слов: если разница длин равна K, то результат алгоритма не может быть меньшим, чем цена пропуска (или вставки) K символов. После такого контроля вычисление ЛМР может уже и не понадобиться.

Применяемые в алгоритме векторы D и I обозначают цены соответственно пропуска и вставки конкретных символов. В

матрице R содержатся цены замены одной на другую всех возможных символов.

Алгоритм ЛМР

Вход $\mathcal{Y} = y_1 y_2 \cdots y_M$ – исправляемое слово,
$\mathcal{X} = x_1 x_2 \cdots x_N$ – исправляющее слово,
Векторы I и D , матрица R.

Выход Локальная мера различия ЛМР$(\mathcal{Y},\mathcal{X})$

Метод A0 $i=1$, $j=1$, $\Lambda MP(0,0)=0$

A1. Если $i>M$ или $j>N$, то
ЛМР$(\mathcal{Y},\mathcal{X})$ = ЛМР$(i-1,j-1)$, переход в A9.

A2. Если $y_i = x_j$, то
ЛМР(i,j) = ЛМР$(i-1,j-1)$, $i=i+1$, $j=j+1$,
переход в A1.

A3. Если $y_i = x_{j+1}$, то
$\Lambda MP(i,j+1) = \Lambda MP(i-1,j-1) + I(x_i)$, $i=i+1$, $j=j+2$,
переход в A1.

A4. Если $y_i \neq x_{j+1}$ или $j=N$, то
$\Lambda MP(i,j) = \Lambda MP(i-1,j-1)$, $i=i+1$.

A5. Если $i>M$ или $j>N$, то
$\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(i-1,j)$, переход в A12.

A6. Если $y_i = x_j$, то
$\Lambda MP(i,j) = \Lambda MP(i-1,j) + D(y_{i-1})$, $i=i+1$, $j=j+1$,
переход в A1.

A7. Если $y_i = x_{j+1}$, то
$\Lambda MP(i,j+1) = \Lambda MP(i-1,j) + R(y_{i-1}, x_j)$, $i=i+1$, $j=j+2$,
переход в A1.

A8. Если $y_i \neq x_{j+1}$ или $j=N$, то
$\Lambda MP(i,j+1) = \Lambda MP(i-1,j) + R(y_{i-1}, x_j)$, $i=i+1$, $j=j+1$,
переход в A5.

A9. Если $i \leq M$, то выполнить A9.1, пока $i \leq M$
A9.1 $\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + D(y_i)$, $i=i+1$,
переход в A15.

A10. Если $j \leq N$, выполнить A10.1, пока $j \leq N$
A10.1 $\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + I(x_j)$, $j=j+1$.

A11. Переход в A15.

A12. Если $i \leq M$, то
$\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + R(y_{i-1}, x_j)$
выполнить A12.1, пока $i \leq M$

AI2.I $\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + D(y_i)$, $i = i+1$
переход в AI5.

AI3. Если $j \leq N$, то
$\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + R(y_{i-1}, x_j)$,
выполнить AI3.I, пока $j \leq N-1$
AI3.I $\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + I(x_{j+1})$, $j = j+1$
переход в AI5.

AI4. Если $i > M$ и $j > N$, то
$\Lambda MP(\mathcal{Y},\mathcal{X}) = \Lambda MP(\mathcal{Y},\mathcal{X}) + D(y_{i-1})$.

AI5. Конец.

5. Примеры

Рассмотрим поведение алгоритма на примерах.
Пусть $\forall a, b \in A \rightarrow D(a) = R(a,b) = I(a) = 1$, где A – алфавит, используемый в сравниваемых словах.

I Пример I
Пусть $\mathcal{Y}$ = ПРИЕН, $\mathcal{X}$ = ПРИМЕР, $(N = M = 6)$
Алгоритм ЛМР работает следующим образом
A0. $i = I$, $j = I$ ЛМР(0, 0) = 0
AI.
A2. ЛМР(I, I) = 0, $i = 2$, $j = 2$
AI.
A4. ЛМР (2, 2) = 0, $i = 3$
A6. ЛМР(3, 2) = 0 + D (П) = I, $i = 4$, $j = 3$
AI.
A2. ЛМР(4, 3) = I, $i = 5$, $j = 4$
AI.
A3. ЛМР(5, 5) = I + I (М) = 2, $i = 6$, $j = 6$
AI.
A4. ЛМР (6, 6) = 2, $i = 7$
A5. ЛМР (ПРИЕМ, ПРИМЕР) = 2
AI2.
AI3. ЛМР (ПРИЕМ, ПРИМЕР) = 2 + R(Н, Р) = 3
AI5. (ЛМР (ПРИЕМ, ПРИМЕР) = 3)

Результат работы алгоритма является равным минимальному расстоянию между словами ПРИЕМ и ПРИМЕР. Рассмотрим поведение алгоритма ЛМР и на другом примере, дающем менее положительный результат.

П Пример 2

Пусть $\mathcal{Y}$ = КИМЕР, $\mathcal{X}$ = ПРИМЕР (N = 5, M = 6)
Алгоритм ЛМР работает следующим образом:
A0. i = I, j = I ЛМР(0, 0) = 0
AI.
A4. ЛМР(I, I) = 0, i = 2
A5.
A8. ЛМР(2, 2) = 0 + R(К, П) = I, i = 3, j = 2
A5.
A8. ЛМР(3, 3) = I + R(И, Р) = 2, i = 4, j = 3
A5.
A8. ЛМР(4, 4) = 2 + R(М, И) = 3, i = 5, j = 4
A5.
A8. ЛМР(5, 5) = 3 + R(Е, М) = 4, i = 6, j = 5
A5. ЛМР(КИМЕР, ПРИМЕР) = 4
AI2.
AI3. ЛМР(КИМЕР, ПРИМЕР) = 4 + R(Р, Е) = 5
AI3.I ЛМР (КИМЕР, ПРИМЕР) = 5 + I(Р) = 6, j = 6
AI5. (ЛМР(КИМЕР, ПРИМЕР) = 6)

Как видно, алгоритм дает трехкратный результат в сравнении с минимальным расстоянием между словами КИМЕР и ПРИМЕР. На этом примере видна выгода повторения работы алгоритма в обратном направлении. Переставим символы в рассмотренных словах в обратном порядке. Получаются $\mathcal{Y}$ = РЕМИК и $\mathcal{X}$ = РЕМИРП. Применение алгоритма на таких $\mathcal{Y}$ и $\mathcal{X}$ дает результат ЛМР (РЕМИК, РЕМИРП) = 2. Окончательное значение ЛМР = min (6, 2) = 2 равно минимальному расстоянию между рассмотренными словами.

Литература

I. Рохтла Х.Х. Каскадные средства обработки синтаксических ошибок. – Всесоюзная конференция по методам трансляции. Тезисы докладов. Новосибирск, 1981, с. 89-91.

2. Рохтла Х.Х. Нейтрализация синтаксических ошибок в системе построения трансляторов ТПИ. – Тр. Таллинск. политехн. ин-та, 1982, № 524, с. 119-129.

3. Рохтла Х.Х. Восстановление синтаксического анализа с помощью синхротроек. – П Всесоюзная конференция: Автоматизация производства пакетов прикладных программ и

трансляторов. Тезисы докладов, Таллин, 1983, с. 144-146.

4. Рохтла Х.Х. Опыт внедрения средств обработки синтаксических ошибок. - Тр. Таллинск. политехн. ин-та, 1983, № 554, с. 71-83.

5. Сидоров А.А. Анализ схожести слов в системе исправления орфографических ошибок. - Программирование, 1979, № 4, с. 65-68.

6. Morgan H.L. Spelling correction in systems programs. - CACM, 1970, N 2, p. 90-94.

7. Wagner R.A. Order-n correction in regular languages. - CACM, 1974, N 5, p. 265-268.

8. Wagner R.A., Fischer M.I. The string-to-string correction problem. - Journal of ACM, 1974, N 1, p. 168-173.

9. Backhouse R.C. Syntax of programming languages. London, Prentice-Hall, 1979. 302 p.

10. Okuda T., Tanaka E., Kasai T. A method for the correction of garbled words based on the Levenshtein metric. - IEEE Transactions on Computers, 1976, N 2, p. 172-177.

11. Tanaka E., Fu K.S. Error correcting parsers for formal languages. - IEEE Transactions on Computers, 1978, N 2, p. 605-616.

12. Neeb F. Ahlichkeitsprüfung von Schutzmarken mit Computer. - Adv. Cybern. and Syst. Res. Proc. Eur. Meet. Vienna, 1972.

H. Rohtla

On a Method for Finding the Similarity of Symbol Sequences

Summary

An algorithm is given to find the similarity of two symbol sequences in linear time. This algorithm is used in syntax error recovery method.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.03

Д.Б. Лийб

ОПТИМИЗАЦИЯ ПАМЯТИ АНАЛИЗАТОРА ГРАММАТИКИ ПРЕДШЕСТВОВАНИЯ РЕДУЦИРУЕМОЙ С (1;1)-ОКК

Эту статью можно рассматривать как продолжение разработки алгоритмов оптимизации анализатора, работающего по методу предшествования, приведенных в [6]. В этой статье был разработан и доказан алгоритм, по которому грамматику предшествования можно было преобразовать так, чтобы для нее всегда существовали две функции предшествования.

Обменивание матрицы предшествования на функции предшествования дает значительную экономию памяти для проведения детектирования. Для грамматик более широкого класса, чем простые грамматики предшествования, такое обменивание ставит ограничения оптимизации памяти для редуцирования.

Задачей данной статьи является преодоление этих ограничений. Для этого определены упорядоченные векторы и дан алгоритм их нахождения. Также описывается алгоритм проведения шага синтаксического анализа - редуцирование при помощи упорядоченных векторов.

Методы оптимизации хорошо применимы для анализатора, работающего с редуцируемой грамматикой предшествования с (m,к)-ограниченным каноническим контекстом (ОКК РГП).

Рассмотрим основные черты вышеназванного анализатора предшествования

1) анализу поддаются все детерминированные языки [5];

2) шаг анализа разбит на два действия

а) детектирование (нахождение основы);

б) редуцирование (нахождение соответствующего основе правила);

3) детектирование производится по методу предшествования [1];

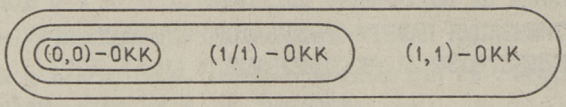
4) редуцирование производится по методу каскада, причем $m,k \leqslant 1$ для (m,k) -ОКК [3];

5) по методу каскада определяется для каждого правила грамматики ее принадлежность к одному из классов грамматик:

а) $K_1 = (0,0)$- ОКК;

б) $K_2 = (I/I)$- ОКК;

в) $K_3 = (I,I)$- ОКК, причем $K_1 \subseteq K_2 \subseteq K_3$. Отношения соответствующих классов грамматик можно представить на диаграмме



Кроме того имеются классы (0,I)-ОКК и (I,0)-ОКК, которые являются составными частями (I/I)-ОКК;

6) для класса K_1 контекст для редуцирования не используется, поскольку между правилами и основами имеется взаимно однозначное отношение;

7) для класса K_2 контекст для редуцирования можно найти непосредственно из матрицы предшествования;

8) для класса K_3 нужна уже специальная таблица контекста.

Такой подход позволяет значительно повысить эффективность анализатора за счет экономии памяти, причем анализатор (I/I)-ОККРГП по эффективности для классов K_1 и K_2 сравним с анализатором простого предшествования.

Далее конкретизируем постановку задачи оптимизации:

Каким способом (при такой же эффективности) сохранить класс K_2, если заменить матрицу предшествования на функции предшествования?

Как известно из [2], при замене матрицы предшествования на функции предшествования часть информации теряется. Если X и Y - два символа грамматики предшествования, то между ними по матрице предшествования могут быть следующие отношения предшествования $X \lessdot Y$, $X \doteq Y$, $X \gtrdot Y$ и $X \not\gtrless Y$, где последнее означает отсутствие отношения предшествования. По функциям предшествования можно восстановить только три первых отношениях предшествования. Но для редупирования правил, входящих в класс K_2, используется так называемый независимый канонический контекст, для определения которого необходимы все четыре отношения предшествования.

Дадим некоторые определения из [5]:

I) контекстно-свободной (КС) грамматикой является четверка $G=\{V_T \cup \{\#\}, V_N, P, S\}$, где V_T - множество терминальных символов, V_N - множество нетерминальных символов и $V_T \cap V_N = \phi$, P - множество правил вида $A \to \alpha$, где $A \in V_N$ и $\alpha \in (V_T \cup V_N)^*$ и $S \in V_N$ является начальной аксиомой;

2) каноническим (m, k)-контекстом для нетерминала A назовем множество $C_A^{(m,k)} = \{(x, y) \mid \#^m S \#^k \overset{*}{\underset{k}{\Rightarrow}} u x A y v, |x| = m, |y| = k, y v \in V_T^* \#^k\}$.

Ограниченным каноническим $(1,1)$-контекстом для нетерминала A назовем множество $C_A^{(1,1)} = \{(X,T) \mid \#S\# \overset{*}{\underset{k}{\Rightarrow}} u X A T_v, T_v \in V_T^* \#\}$;

3) необходимым условием редуцируемости (1,1)-ОККРГП является $C_A^{(1,1)} \cap C_B^{(1,1)} = \phi$, причем $B \in P_\alpha^A$, $P_\alpha^A = \{B \mid B \in P_\alpha \& B \neq A\}$,

$$P_\alpha = \{A \mid A \to \alpha \in P\};$$

4) $$LC(A) = \{X \mid (X, \lambda) \in C_A^{(1,0)}\},$$

$$RC(A) = \{T \mid (\lambda, T) \in C_A^{(0,1)}\},$$

$$C_A^{(1/1)} = LC(A) \times RC(A). \qquad (I)$$

Методы оптимизации памяти

Обозначим через $C_{A,\alpha}$ подмножество пар контекста для класса грамматик предшествования K_3.

Если грамматика предшествования (1,1)-ОККРГП и имеет место условие $C_A^{(1/1)} \cap C_B^{(1/1)} \neq \phi$, тогда по (I) можно записать

$$[LC(A) \times RC(A)] \cap [LC(B) \times RC(B)] \neq \phi, \quad B \in P_\alpha^A.$$

Нетрудно догадаться, что это условие справедливо, если существуют $X_1 \in LC(A)$, $X_2 \in LC(B)$, $T_1 \in RC(A)$, $T_2 \in RC(B)$, такие что $(X_1 = X_2) \& (T_1 = T_2)$.

Таким образом,

$$C_{A,\alpha} = \bigcup_{B \in P_\alpha^A} (C_A^{(1,1)} \cap C_B^{(1,1)}) = \{(X,T) \mid X \in LC(A) \cap LC(B) \& T \in RC(A) \cap RC(B)\}$$

По существу множество $C_{A,\alpha}$ содержит пары контекста, которые должны сохраняться для проведения синтаксического анализа, для остальных анализ проводится согласно классу K_2.

Пример I. Приведем КС-грамматику, в которой имеются правила, принадлежащие к классам K_1, K_2, K_3 и множествам контекстных пар $C_{A,\alpha}$.

$G=\{V_T, V_N, P, S\}$, $V_T=\{a,b,c,d,e,f\}$, $V_N=\{S,A,B,D,E,F,H\}$,

$P=\{S\to aAC,\ S\to bAd,\ S\to aBd,\ S\to bDc,$

$A\to e,\ B\to e,\ D\to e$

$S\to aEb,\ S\to aFa,\ S\to bFa,\ S\to bHb$

$E\to f,\ F\to f,\ H\to f\}$.

Пусть $P=P_1\cup P_2\cup P_3$ так, что $P_1\in K_1$, $P_2\in K_2$, $P_3\in K_3$

тогда $P_1=\{S\to aAc,\ S\to bAd,\ S\to aBd,\ S\to bDc,$

$S\to aEb,\ S\to aFa,\ S\to bFa,\ S\to bHb\}$,

$P_2=\{E\to f,\ F\to f,\ H\to f\}$,

$P_3=\{A\to e,\ B\to e,\ D\to e\}$

причем 1) $C_A^{(1/1)}=\{(a,c),(b,d),(a,d),(b,c)\}$, $C_B^{(1/1)}=\{(a,d)\}$ $C_D^{(1/1)}=\{(b,c)\}$

2) $C_A^{(1/1)}=\{(a,c),(b,d)\}$, $C_B^{(1,1)}=\{(a,d)\}$, $C_D^{(1,1)}=\{(b,c)\}$

3) $C_{A,e}=\phi$ $C_{B,e}=\{(a,d)\}$. $C_{D,e}=\{(b,c)\}$

4) $C_E^{(1/1)}=\{(a,b)\}$, $C_F^{(1/1)}=\{(a,a),(a,b)\}$, $C_H^{(1/1)}=\{(b,b)\}$, $C_E^{(1/1)}\cap C_F^{(1/1)}=\phi$.

Таким образом, вместо четырех пар контекста, приведенных в 2), надо явно сохранить только два (согласно 3)).

Далее рассмотрим метод, по которому информация для редуцирования класса K_2 формируется в упорядоченные векторы.

Для этого надо найти рефлексивно-линейное упорядочение между символами контекста для классов K_2 и K_3.

С этой целью рассмотрим КС-грамматику, где $P=P_1\cup P_2$, $P_1\in K_1$ и $P_2\in(K_2\cup K_3)$, причем $P_2=\bigcup\limits_{\alpha\in\Theta}P_\alpha$, $\Theta=\{\alpha\,||P_\alpha|>1\}$.

Также определим множество символов грамматики

$$\mathfrak{M}=\{a\,|\,a\in LC(A)\,\&\,A\in P_2\}\cup\{a\,|\,\exists B\exists\alpha(B\in P_\alpha^A\,\&\,LC(B)\cap LC(A)\neq\phi\,\&\,a\in RC(A))\}$$

для которых существует бинарное отношение $\gamma=\{(a,B)\,|\,a\in\mathfrak{M},\ B\in P_\alpha^A\}$. Обозначим $\gamma B=\{a\,|\,a\,\gamma\,B\}$, $a\gamma=\{B\,|\,a\,\gamma\,B\}$. Если $a_i\in\gamma B\,\&\,a_j\in\gamma C$ и при этом $i\leq j$ для каждого $B,C\in P_\alpha^A$, можно говорить, что существует рефлексивно-линейное упорядочение для отношения γ.

Отношение γ для P_2 можно представить в виде массива LR, в котором строки обозначают контекстные символы и

столбцы - соответствующие нетерминальные символы, принадлежащие к P_α^A. Элемент массива $LR_{ij}=1$, если $a_i \gamma B_j$, в противном случае $LR_{ij}=0$.

Рефлексивно-линейное упорядочение такого массива может быть описано алгоритмом, работающим по методу перебора с возвратом, при помощи которого происходит замена строк i и j между собой. Замена строк i и j производится, если строка i нарушает и строка j не нарушает рефлексивно-линейное упорядочение хотя бы в одном $P_\alpha \in P_2$ для строк меньше i.

Такая замена требует перенумерации внешних кодов символов грамматики, при этом грамматика не изменяется.

Несложно вывести ситуации, когда для P_α не находится рефлексивно-линейной упорядоченности. Это может иметь место, если число элементов подмножества $|P_\alpha| \geq 3$ и $a_{i\gamma} = \{B,C\}$, $a_{j\gamma}=\{B,D\}$, $a_{k\gamma}=\{C,D\}$ и $i \neq j \neq k$, $B,C,D \in P_\alpha^A$.

Чтобы обойти такие ситуации, можно преобразовать грамматику при помощи частных видов стратификации, приведенных в [7]:

I) если $a \in \gamma A$ соответствует $a \doteq A$ или $A \doteq a$, тогда для извлечения $a \gamma A$ применять стратификацию направо;

2) если $a \in \gamma A$ соответствует $a \lessdot A$ или $A \lessdot a$, тогда для извлечения $a \gamma A$ применять полную стратификацию налево.

При выполнении стратификации по возможности желательно выбрать такие a и A, чтобы вследствие стратификации между ними не образовалось новое множество P_α.

Далее приведем блок-схему алгоритма синтаксического анализа для (1,1)-ОККРГП шага редуцирования (см. фиг. I). Этот алгоритм подходит, если редуцирование проводится при помощи

а) матрицы предшествования;
б) упорядоченных векторов.

На этой блок-схеме видны два выхода. Выбор входа зависит от P_α, входящего либо в K_2, либо в K_3.

Нас интересуют только действия, связанные с условиями и обозначенные на схеме номерами I), 2), 3). Рассмотрим эти условия по очередности:

I) исследуем, принадлежит ли заданная пара контекста (a,b) ко множеству $C_{B,\alpha}$. Контекстные пары хранятся при этом в хэш-таблице;

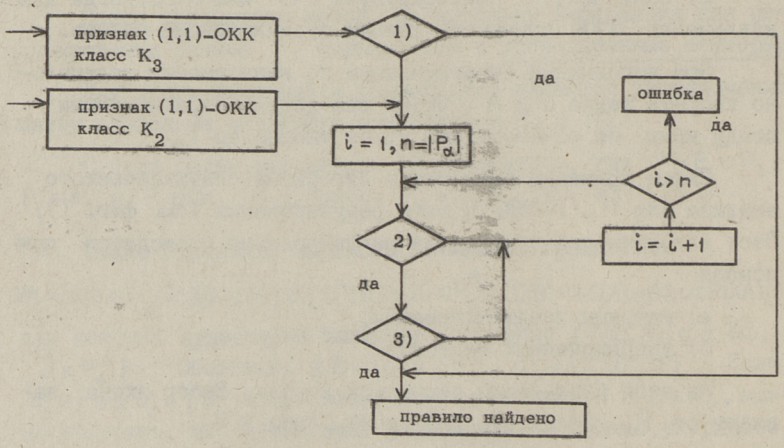
2) по заданной паре исследуют, принадлежит ли (a,λ) ко множеству $C_B^{(1,0)}$. При проверке с помощью матрицы предшествования это условие истинно, если $a \lessdot B$ или $a \doteq B$. При проверке условия с помощью упорядоченных векторов исследуют, принадлежит ли в указанный промежуток $K_1 \leq a \leq K_2$ для левого контекста;

3) по заданной паре (a,b) исследуют, принадлежит ли (λ,b) ко множеству $C_B^{(0,1)}$. При проверке с помощью матрицы предшествования это условие истинно, если $B \lessdot b$, $B \doteq b$ или $B \gtrdot b$. При проверке с помощью упорядоченных векторов есть две возможности

а) если промежуток $K_1 = K_2 = 0$ для правого контекста, тогда достаточно, если удовлетворено условие в 2) для $e_B^{(1,0)}$ и B является искомым нетерминалом;

б) если $K_1 \leq b \leq K_2$, тогда пара принадлежит множеству и B является искомым нетерминалом.

Если для $B \in P_\alpha^A$ не удовлетворено ни одно условие, тогда имеется ошибка редуцирования.



Фиг. 1. Блок-схема шага редуцирования синтаксического анализа.

Как видно, алгоритм в обоих случаях по существу одинаков. Разница только в норме представления информации, необходимой для редуцирования.

Пример 2. На этом примере будет показано, как можно формировать упорядоченные векторы. Для примера используем грамматику, приведенную в примере I.

	левый контекст		правый контекст	
нетерм. / конт. символ	A B D	E F H	A B D	E F H
a	1 1	1 1		1
b	1 1	1 1		1 1
c			1 1	
d			1 1	
	$\alpha = e$	$\alpha = f$	$\alpha = c$	$\alpha = f$

Эта таблица соответствует массиву LR. Здесь для каждого P_α уже имеется рефлексивно-линейное упорядочение, причем $a < b < c < d$. Выпишем далее правила и промежутки контекстных символов, соответствующих упорядоченным векторам.

	левый контекст		правый контекст	
промежутки / правила	K_1	K_2	K_1	K_2
A → e	a	b	d	c
B → e	a	a	b	b
D → e	b	b	d	d
E → f	a	a	b	b
F → f	a	b	a	a
H → e	b	b	b	b

На конкретном шаге анализа выбирается правило (из соответствующего P_2), удовлетворяющее промежуток контекста упорядоченных векторов.

Практические результаты исследования

Выше предлагаемый метод оптимизации синтаксического анализатора, работающего на грамматиках предшествования, редуцируемых с (I,I)-ОКК, вполне оправдывает себя.

Такая уверенность появилась при внедрении метода в систему построения трансляторов ELMA в Таллинском политехническом институте [4].

Рефлексивно-линейное упорядочение вычислялось на более чем двадцати грамматиках, описанных при помощи СПТ ELMA. В том числе были различные грамматики для проблемно-ориентированных языков и подмножества языков программирования (C, FORTRAN, ADA). Число правил при этом менялось в промежутке от 25 до 700. Только в одной грамматике (подмножество ADA) нашлась одна ситуация, когда без преобразований невозможно было добиться рефлексивно-линейного упорядочения. Стратификации в правилах грамматики не усложняют алгоритм редуцирования, увеличивается только число правил.

Объем памяти для сохранения информации, позволяющий произвести редуцирование при помощи упорядоченных векторов, можно вычислить следующим образом: $|P_2| * 4$ байта, т.е. для каждого правила разбора грамматики, имеющего одинаковую правую часть с каким-то другим правилом, выделяется 4 байта для представления двух промежутков. Первые два - K_1 и K_2 - для левого контекста, и остальные два - K_1 и K_2 - для правого контекста.

Зависимость между числами элементов множеств $|P|$ и $|P_2|$ для различных грамматик приведена в следующей таблице

$\|P\|$	22	45	54	75	98	I20	I53	228	52I	672
$\|P_2\|$	3	2	22	I4	I6	I4	28	40	66	89

Литература

I. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Том I. Синтаксический анализ. М., Мир, I978. 6I6 с.

2. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Том 2. Компиляция. М., Мир, I978. 486 с.

3. Вооглайд А.О., Лепп М.В. Опыт внедрения классических методов синтаксического анализа. - Тр. Таллинск. политехн. ин-та, I979, № 464, с. 3-20.

4. В о о г л а й д А.О., Л е п п М.В., Л и й б Д.Б. Входные языки системы ELMA.- Тр.Таллинск. политехн, ин-та, 1982, № 524, с. 79-96.

5. В о о г л а й д А.О., Т о м б а к М.О. О проблемах редуцирования в грамматиках предшествования. - Тр. Таллинск. политехн. ин-та, 1975, № 386, с. 23-37.

6. Л и й б Д.Б. О преобразовании матрицы предшествования в векторы предшествования. - Тр. Таллинск. политехн. ин-та, 1983, № 554, с. 95-109.

7. Т о м б а к М.О. Об устранении конфликтов предшествования. - Тр. ВЦ Тартуского гос. ун-та, Тарту, 1976, № 37, с. 60-91.

D. Liib

Optimization of Memory Requirements for Precedence Grammars using (I,I)-Bounded Canonical Context Reduction

S u m m a r y

In this article a new method to optimize memory requirements for precedence grammars is described. The economy results as a transformation of reduction information into ordered vectors.

This method suits especially when a precedence matrix is represented by precedence functions.

№ 568

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 519.4

П.А. Пуусемп

АБСТРАКТНАЯ ХАРАКТЕРИСТИКА ГРУПП ШМИДТА ПО ИХ ПОЛУГРУППАМ ЭНДОМОРФИЗМОВ

I. Введение

Конечная ненильпотентная группа называется группой Шмидта, если все ее собственные подгруппы нильпотентны. Строение групп Шмидта хорошо известно (см. например, работы Л. Редей [10, 11]). В работах [2, 3] дается описание полугрупп эндоморфизмов и групп автоморфизмов групп Шмидта. В настоящей работе мы докажем, что класс групп Шмидта определяется их полугруппами эндоморфизмов, т.е. если полугруппа всех эндоморфизмов группы G изоморфна полугруппе всех эндоморфизмов некоторой группы Шмидта, то группа G также является группой Шмидта.

Будем придерживаться следующих обозначений:

$[g,h] = g^{-1}h^{-1}gh$;

$J_0(G)$ – совокупность всех неединичных и ненулевых идемпотентных эндоморфизмов группы G;

$\mathrm{End}G$ – полугруппа всех эндоморфизмов группы G;

$\mathrm{Aut}G$ – группа всех автоморфизмов группы G;

$\hat{g}$ – внутренний автоморфизм, порожденный элементом g;

$[x] = \{y \in \mathrm{End}G \mid y^2 = y;\ xy = y;\ yx = x\}$,

где x – идемпотент полугруппы $\mathrm{End}G$;

$$K_G(x) = \{y \in \mathrm{End}G \mid yx = xy = y\};$$

$$V_G(x) = \{y \in \mathrm{Aut}G \mid yx = x\};$$

$$D_G(x) = \{y \in \mathrm{Aut}G \mid yx = xy = x\};$$

$$H_G(x) = \{y \in \mathrm{End}G \mid xy = y,\ yx = 0\}.$$

2. Вспомогательные результаты

Перечислим основные свойства групп Шмидта (см. работы Л. Редей [10, 11]). Каждая группа Шмидта характеризуется тремя параметрами p, q и ν, где p, q - различные простые числа и ν - натуральное число. Заданным параметрам соответствует, вообще говоря, много групп Шмидта.

Пусть всюду в этом пункте G - некоторая группа Шмидта с параметрами p, q и ν. Тогда: 1) группа G разлагается в полупрямое произведение $G = G' \lambda C_{q^\nu}$ своего коммутанта G' и циклической подгруппы $C_{q^\nu} = \langle b \rangle$ порядка q^ν; 2) коммутант G' является p-группой; 3) второй коммутант G'' группы G содержится в центре $Z(G)$ группы G; 4) факторгруппа G'/G'' является элементарной абелевой p-группой порядка p^u (число u определяется позже); 5) $Z(G/G'') = \langle b^q G'' \rangle$.

Обозначим $G^* = G/G''$. Группа G^* является непримарной группой Миллера-Морено. Напомним, что конечную некоммутативную группу называют группой Миллера-Морено, если все ее собственные подгруппы коммутативны. Непримарные группы Миллера-Морено определяются однозначно с точностью изоморфизма с параметрами p, q и ν. Следуя работе [2], перечислим некоторые свойства полугрупп всех эндоморфизмов групп G и G^*.

Естественное отображение $\tau \mapsto \tau^*$ из полугруппы $\mathrm{End}\, G$ в полугруппу $\mathrm{End}\, G^*$, определяемое правилом $(gG'')\tau^* = (g\tau)G''$ $(g \in G, \tau \in \mathrm{End}\, G)$, является мономорфизмом, который порождает изоморфизм между полугруппами всех собственных эндоморфизмов групп G и G^*.

Пусть $\psi(x)$ - произвольный неприводимый делитель (нормированный) многочлена

$$\frac{x^{q}-1}{x-1} = x^{q-1} + x^{q-2} + \cdots + x + 1 \in Z_p[x],$$

где $Z_p[x]$ - кольцо многочленов над полем вычетов Z_p по простому числу p.

Обозначим через $\bar{Z}_p[x]$ факторкольцо кольца $Z_p[x]$ по главному идеалу, порожденному многочленом $\psi(x)$. Тогда собственные эндоморфизмы группы G^* задаются парами $[n; f(x)]$, где $n \in Z_{q^\nu}$ и $f(x) \in \bar{Z}_p[x]$. При этом две пары $[n_1; f_1(x)]$ и $[n_2; f_2(x)]$ равны тогда и только тогда, когда I) $n_1 =$

$= n_2 (=n)$; 2) $f_1(x)=f_2(x)$ или число n делится на q. Автоморфизмы группы G^* задаются тройками $[n; a(x); b(x)]$, где $n\in Z_{q^\nu}$; $a(x), b(x)\in \bar{Z}_p[x]$; $b(x)\neq 0$ и число n не делится на q. Различным тройкам соответствуют различные автоморфизмы. При этом

$$[n; f(x)]\cdot[m; g(x)] = [nm; g(x)],$$
$$[n; a(x); b(x)]\cdot[m; f(x)] = [nm; f(x)], \qquad (2.I)$$
$$[m; f(x)]\cdot[n; a(x); b(x)] = [nm; a(x)\cdot(x-1)^{-1} + b(x)\cdot f(x^n)].$$

3. Основные теоремы

Обозначим через u степень неприводимого многочлена $\psi(x)$, упомянутого в предыдущем пункте. Число u равняется порядку элемента p в группе обратимых элементов кольца Z_q.

Теорема 3.I. Для того, чтобы конечная группа G была изоморфна группе Шмидта с параметрами p, q и ν, необходимо и достаточно, чтобы существовал $x\in J_0(G)$, удовлетворяющий условиям:

I) $K_G(x) \simeq \mathrm{End}\, C_{q^\nu}$;

2) $H_G(x) = \{0\}$;

3) $J_0(G) = [x]$;

4) $|J_0(G)| = p^u$;

5) $z \in \bigcap_{y\in J_0(G)} K_G(y)$ тогда и только тогда, когда $z^{\nu} = 0$;

6) $\mathrm{End}\, G \setminus \mathrm{Aut}\, G = \bigcup_{y\in J_0(G)} K_G(y)$;

7) $D_G(x)$ является p'-подгруппой группы $V_G(x)$;

8) силовская p-подгруппа группы $V_G(x)$ является элементарной абелевой группой порядка p^u.

Доказательство необходимости для теоремы 3.I. Пусть G – группа Шмидта с параметрами p, q и ν. Тогда

$$G = G' \lambda \langle b \rangle, \qquad (3.I)$$

где $\langle b \rangle \simeq C_{q^\nu}$.

Обозначим через x идемпотент, соответствующий полупрямому разложению (3.I), т.е. $\mathrm{Im}\, x = \langle b \rangle$, $\mathrm{Ker}\, x = G'$. Так как $K_G(x) \simeq \mathrm{End}(\mathrm{Im}\, x)$ (см. [4], лемма I.6), то условие I) имеет место.

Пусть $z \in H_G(x)$. Тогда $zx=0$ и $xz=z$. Поэтому $Im\, z \subset Ker\, x = G', Ker\, x \subset Ker\, z$ и $bz \in G'$. Отсюда следует, что $bz=1$, ибо b – q-элемент и G' – p-группа. Следовательно, $z=0$ и $H_G(x)=\{0\}$. Справедливость условия 2) доказана.

Поскольку мономорфизм $*$ порождает изоморфизм между полугруппами собственных эндоморфизмов групп G и $G^*=G/G''$, то достаточно доказать свойства 3)–6) при $G=G^*$ и $x=x^*$. Докажем эти свойства.

По формулам (2.I) ясно, что $[0; f(x)]$ является нулевым элементом полугруппы $End\, G^*$ и ненулевыми собственными идемпотентами являются лишь эндоморфизмы $[1; f(x)]$, т.е.

$$J_0(G^*) = \{[1; f(x)] \mid f(x) \in \bar{Z}_p[x]\}. \qquad (3.2)$$

В силу $[1; f(x)]\cdot[1; g(x)] = [1; g(x)]$ ясно, что $J_0(G^*)=[x^*]$, т.е. справедливо свойство 3).

Идемпотенты $[1; f(x)]$ и $[1; g(x)]$ равны тогда и только тогда, когда $f(x)=g(x)$. Поэтому $|J_0(G^*)| = |\bar{Z}_p[x]| = p^{\omega}$ и имеет место свойство 4).

По формулам (2.I) получаем равенства

$$[n; f(x)]\cdot[1; f(x)] = [1; f(x)]\cdot[n; f(x)] = [n; f(x)],$$

т.е. $[n; f(x)] \in K_{G^*}([1; f(x)])$. Отсюда следует справедливость равенства 6).

Предположим, что

$$[n; g(x)] \in \bigcap_{y \in J_0(G^*)} K_{G^*}(y).$$

Тогда по равенству (3.2) $[n; g(x)] \in K_{G^*}([1; f(x)])$ для каждого $f(x) \in \bar{Z}_p[x]$. Отсюда

$$[n; g(x)]\cdot[1; f(x)] = [1; f(x)]\cdot[n; g(x)] = [n; g(x)].$$

Так как первое произведение в последних равенствах равно $[n; f(x)]$, то $[n; g(x)]=[n; f(x)]$ при каждом $f(x) \in \bar{Z}_p[x]$. Это равносильно тому, что число n делится на q. Следовательно,

$$\bigcap_{y \in J_0(G^*)} K_{G^*}(y) = \{[n; 0] \mid n \in q\cdot Z_{q^{\nu}}\}. \qquad (3.3)$$

Из равенства (3.3) сразу следует, что

$$z = [n; f(x)] \in \bigcap_{y \in J_0(G^*)} K_{G^*}(y)$$

тогда и только тогда, когда $z^{\nu} = [n^{\nu}; f(x)] = 0 = [0; f(x)]$. Справедливость условия 5) доказана.

Докажем теперь условия 7) и 8). Определим сначала $|V_{G^*}(x^*)|$. Пусть $z \in \mathrm{Aut}G^*$. Тогда z имеет вид $z=[n; a(x); b(x)]$, где $b(x) \neq 0$ и число n не делится на q. В работе [2] установлено, что $x^*=[1;0]$. Автоморфизм z принадлежит группе $V_{G^*}(x^*)$ тогда и только тогда, когда $z \cdot x^* = x^*$, т.е.

$$[n; a(x); b(x)] \cdot [1;0] = [n;0] = [1;0].$$

Это означает, что $n=1$ и

$$V_{G^*}(x^*) = \{[1; a(x); b(x)] \mid a(x), b(x) \in \bar{Z}_p[x]; b(x) \neq 0\}.$$

Следовательно,

$$|V_{G^*}(x^*)| = p^u(p^u-1), \tag{3.4}$$

ибо $|\bar{Z}_p(x)| = p^u$.

Вычислим еще $|D_{G^*}(x^*)|$. Группа $D_{G^*}(x^*)$ состоит из таких $z = [1; a(x); b(x)] \in V_{G^*}(x^*)$, для которых $x^* \cdot z = x^*$, т.е.

$$[1;0]\,[1; a(x); b(x)] = [1; \frac{a(x)}{x-1}] = [1;0],$$

откуда $a(x)=0$. Поэтому

$$|D_{G^*}(x^*)| = p^u - 1. \tag{3.5}$$

Явно $(D_G(x))^* \subset D_{G^*}(x^*)$. Условие 7) вытекает теперь из равенства (3.5).

Ранее было отмечено, что $G'' \subset Z(G)$, $Z(G/G'')$ - q-группа и G'/G'' - элементарная абелева p-группа порядка p^u. Поэтому $G'' \subset G' \cap Z(G)$ и

$$Z(G/G'') \cap (G'/G'') = \langle 1 \rangle. \tag{3.6}$$

Если $g \in G' \cap Z(G)$, то $gG'' \in Z(G/G'') \cap (G'/G'')$ и в силу равенства (3.6) $gG'' = G''$ и $g \in G''$. Следовательно, $G' \cap Z(G) \subset G''$, $G'' = G' \cap Z(G)$ и

$$\hat{G}' = \{\hat{g} \mid g \in G'\} \simeq G'/(G' \cap Z(G)) = G'/G''.$$

Теперь получено, что $\hat{G}'$ (а также $(\hat{G}')^*$) является элементарной абелевой группой порядка p^u. По лемме 4 из [5] $\hat{G}' \subset V_G(x)$. Но тогда также $(\hat{G}')^* \subset V_{G^*}(x^*)$. Из равенства (3.4) теперь следует, что $(\hat{G}')^*$ является силовской p-подгруппой группы $V_{G^*}(x^*)$. Поэтому $\hat{G}'$ является силовской p-подгруппой группы $V_G(x)$. Отсюда следует справедливость условия 8). Необходимость условий I)-8) в теореме 3.I доказана.

Для доказательства достаточности условий I)-8) в теореме 3.I докажем сначала ряд лемм.

Лемма 3.2. Если x, y - идемпотенты полугруппы $\mathrm{End}G$, то $y \in [x]$ тогда и только тогда, тогда, $\mathrm{Ker}x = \mathrm{Ker}y$.

Доказательство. Если $y \in [x]$, то $xy = y$, $yx = x$ и поэтому $\mathrm{Ker}\,x \subset \mathrm{Ker}\,y$, $\mathrm{Ker}\,y \subset \mathrm{Ker}\,x$, т.е. $\mathrm{Ker}\,x = \mathrm{Ker}\,y$. Пусть, наоборот, $\mathrm{Ker}\,x = \mathrm{Ker}\,y$. По лемме 3.1 из [4] каждый $g \in G$ имеет вид $g = kh$, где $k \in \mathrm{Im}\,x$, $h \in \mathrm{Ker}\,x$, $kx = k$. Поэтому $g(xy) = ky = (kh)y = gy$, т.е. $xy = y$. Аналогично получаем $yx = x$. Следовательно, $y \in [x]$. Лемма доказана.

В следующих леммах будем считать, что G – конечная группа и x – ее идемпотентный эндоморфизм, удовлетворяющий условиям 1)–8) теоремы 3.1.

Лемма 3.3. Пусть S – произвольная силовская p-подгруппа группы G. Тогда найдется такой $b \in G$, что имеют место формулы

$$G = \mathrm{Ker}\,x \lambda \mathrm{Im}\,x, \quad \mathrm{Im}\,x = \langle b \rangle \simeq C_{q^\nu}, \tag{3.7}$$

$$J_0(G) = [x] = \{ x\hat{g} \mid g \in G \}, \tag{3.8}$$

$$J_0(G) = \{ x\hat{g} \mid g \in \mathrm{Ker}\,x \}, \tag{3.9}$$

$$[\mathrm{Ker}\,x : C_{\mathrm{Ker}\,x}(b)] = p^u = [S : C_S(b)], \tag{3.10}$$

$$J_0(G) = \{ x \cdot \hat{s} \mid s \in S \}. \tag{3.11}$$

При этом все q'-элементы группы G содержатся в $\mathrm{Ker}\,x$ и $\mathrm{Im}\,x = \langle b \rangle$ является силовской q-подгруппой в G.

Доказательство. По лемме 3.1 из [4] $G = \mathrm{Ker}\,x \lambda \mathrm{Im}\,x$. По лемме 3.6 из [4] и условию 1) $\mathrm{End}(\mathrm{Im}\,x) \simeq \mathrm{End}\,C_{q^\nu}$. Так как каждая конечная абелева группа определяется своей полугруппой эндоморфизмов (см. [4], теорема 4.2), то $\mathrm{Im}\,x \simeq C_{q^\nu}$ и существует такой $b \in G$, что $\mathrm{Im}\,x = \langle b \rangle$.

Из условия 2) следует, что $\mathrm{Ker}\,x$ является q'-группой. Действительно, в противном случае группа $\mathrm{Ker}\,x$ содержит элемент g порядка q и существует ненулевой z из $H_q(x) : (\mathrm{Ker}\,x)z = \langle 1 \rangle$, $bz = g$. Следовательно, $\langle b \rangle$ является силовской q-подгруппой группы G. Так как $\mathrm{Ker}\,x \Delta G$ и все силовские подгруппы по одному и тому же простому числу сопряжены между собой, то все q'-элементы содержатся в $\mathrm{Ker}\,x$.

Для доказательства формул (3.8) и (3.9) учтем, что по условию 3) будет $J_0(G) = [x]$. Поэтому для получения этих формул достаточно вывести включения $[x] \subset \{ x\hat{g} \mid g \in \mathrm{Ker}\,x \}$ и $\{ x\hat{g} \mid g \in G \} \subset J_0(G)$. Для вывода первого включения заметим, что если $y \in [x]$, то по лемме 3.2 $\mathrm{Ker}\,x = \mathrm{Ker}\,y$ и поэтому $\mathrm{Im}\,x = \langle b \rangle \simeq \mathrm{Im}\,y$.

Поскольку $Jm\varkappa$ – силовская q-подгруппа группы G, то существует такой $g\in G$, что $Jm\gamma=(Jm\varkappa)\hat{g}=G(x\hat{g})$. В силу $Ker\gamma = Ker\varkappa = (Ker\varkappa)\hat{g} = Ker(x\hat{g})$ ясно, что $\gamma = x\hat{g}$. Первое включение получено.

Пусть $h,g\in G$. Так как $Jm\varkappa$ коммутативна, то коммутатор $[h\varkappa,g]$ содержится в $Ker\varkappa$ и

$$h(x\hat{g})^2=(g^{-1}\cdot h\varkappa\cdot g)(x\hat{g})=(h\varkappa\cdot[h\varkappa,g])(x\hat{g})=$$
$$=(h\varkappa^2)\hat{g}=h(x\hat{g}),$$

т.е. $(x\hat{g})^2=x\hat{g}\in J_0(G)$. Включение $\{x\hat{g}\mid g\in G\}\subset J_0(G)$ также доказано. Этим доказаны равенства (3.8) и (3.9).

Чтобы получить первое равенство из (3.I0), подсчитаем $|J_0(G)|$ из равенства (3.9). Предположим, что $g,h\in Ker\varkappa$ и $x\hat{g}=x\hat{h}$. Тогда в силу $Jm\varkappa=\langle b\rangle$ имеем $gh^{-1}\in C_{Ker\varkappa}(b)$. Поэтому количество различных элементов вида $x\hat{g}$, $g\in Ker\varkappa$, равно $[Ker\varkappa:C_{Ker\varkappa}(b)]$, откуда в силу равенства (3.9) и условия 4) получим первое равенство из (3.I0).

Для вывода (3.II) и второго равенства из (3.I0) заметим, что $S\subset Ker\varkappa$. Ясно, что $[S:C_S(b)]\le[Ker\varkappa:C_{Ker\varkappa}(b)]=p^u$. Пусть S_0 – силовская p-подгруппа группы $C_{Ker\varkappa}(b)$, содержащая группу $C_S(b)$. Тогда $|C_{Ker\varkappa}(b)|=|S_0|\cdot t$, где число t не делится на p. Согласно первому равенству из (3.I0) имеем $|Ker\varkappa| = p^u\cdot|C_{Ker\varkappa}(b)|=p^u\cdot|S_0|\cdot t$. Поэтому $|S|=p^u\cdot|S_0|$. Если $[S:C_S(b)]<p^u$, то $|C_S(b)|>|S|:p^u=|S_0|$, что невозможно, ибо $S_0\supset C_S(b)$. Следовательно, $[S:C_S(b)]=p^u$. Отсюда следует и равенство (3.II). Лемма доказана.

В дальнейшем мы будем считать, что S – произвольная силовская p-подгруппа группы G и b – элемент, для которого $Jm\varkappa=\langle b\rangle$.

Лемма 3.4. Имеют место формулы

$$C_S(b)\subset Z(G), \tag{3.I2}$$

$$b^{\vartheta}\in Z(G). \tag{3.I3}$$

Доказательство. Для проверки (3.I2) заметим, что по лемме 6 из [6] $\widehat{C_S(b)}=\{\hat{s}\mid s\in C_S(b)\}\subset D_G(\varkappa)$. При этом $\widehat{C_S(b)}$ – p-группа, ибо S – p-группа. Согласно условию 7) $D_G(\varkappa)$ является p'-группой. Поэтому $\widehat{C_S(b)}=\langle 1\rangle$ и $C_S(b)\subset Z(G)$.

Переходим к установлению того, что для любого $g\in G$ будет $[b^{\vartheta},g]=1$. Берем сначала произвольный $g\in G$, обозначим

$y=x\hat{g}$ и заметим, что по формуле (3.8) $y\in J_0(G)$. Определим теперь отображение z :

$$hz=\begin{cases} h^{g}=g^{-1}b^{g}g, & \text{если } h=g^{-1}bg\in \operatorname{Im} y,\\ 1 & \text{если } h\in \operatorname{Ker} y.\end{cases}$$

Так как $G=\operatorname{Ker} y\lambda \operatorname{Im} y$, то продолжив z по формуле $(uv)z=(uz)(vz)$ на все G, мы получим эндоморфизм группы G.

Покажем, что $bz\in\langle b\rangle=\operatorname{Im} x$, $G'\subset\operatorname{Ker} x$. Из определения z следует, что $hz^{r}=g^{-1}b^{g^{r}}g=1$. Поэтому $z^{r}=0$ и в силу условия 5) $z\in\bigcap_{u\in J_0(G)}K_G(u)$. Это влечет $z\in K_G(x)$. По лемме I.5 из [4] теперь $bz\in\langle b\rangle$, т.е. $bz=b^{n}$ для некоторого целого числа n. По определению z группа $\operatorname{Im} z\simeq G/\operatorname{Ker} z$ коммутативна. Поэтому $G'\subset\operatorname{Ker} z$ и в частности, $[b,g]z=1$.

Для доказательства равенства (3.I3) вычислим двумя способами bz :

$$b^{n}=bz=(bz)([b,g]z)=(b[b,g])z=$$
$$=(g^{-1}bg)z=g^{-1}b^{g}g=b^{g}[b^{g},g].$$

Теперь мы заключаем из включения $G'\subset\operatorname{Ker} x$, что $[b^{g},g]=b^{n-g}\in\operatorname{Im} x\cap\operatorname{Ker} x$ и равенство (3.7) дает $[b^{g},g]=1$. Поскольку g – произвольный элемент группы G, то $b^{g}\in Z(G)$. Лемма доказана.

Лемма 3.5. Существует такая силовская p-подгруппа S группы G, для которой $b\in N_G(S)$ и $\langle b,S\rangle=S\lambda\langle b\rangle$.

Доказательство. Пусть S_1 – произвольная силовская p-подгруппа группы G. Обозначим $N=N_G(S_1)$. В силу (3.I3) имеем $\langle b^{g}\rangle\subset N$, откуда следует, что число $|N|$ делится на $q_i^{\nu-1}$. Если $|N|$ не делится на q_i^{ν}, то $\langle b^{g}\rangle$ является силовской q_i-подгруппой группы N. При этом согласно лемме 3.4 $\langle b^{g}\rangle\subset Z(N)$. Поэтому $N=\langle b^{g}\rangle\times N_1$, где N_1 – холловская q_i'-подгруппа группы N. В силу леммы 3.3 $N_1\subset\operatorname{Ker} x$ и $N\subset\langle\operatorname{Ker} x,b^{g}\rangle$. По теореме 4.2.4 из [7] подгруппа $\langle\operatorname{Ker} x,b^{g}\rangle$ совпадает со своим нормализатором. Но $\langle\operatorname{Ker} x,b^{g}\rangle\lhd G$. Поэтому $G=\langle\operatorname{Ker} x,b^{g}\rangle$. Это противоречит равенству (3.7). Следовательно, число $|N|$ делится на q_i^{ν}.

Для построения подгруппы S заметим, что согласно лемме 3.3 подгруппа $\langle b\rangle$ является силовской q_i-подгруппой порядка q_i^{ν} в G. По первой части доказательства одна из силовских q_i-подгрупп группы G содержится в N. Поэтому существует такой $g\in G$, что $g^{-1}\cdot\langle b\rangle\cdot g\subset N=N_G(S_1)$. Тогда $\langle b\rangle\subset N_G(gS_1g^{-1})$.

Обозначим $S=gS_1g^{-1}$. По лемме 3.3 $S\subset \mathrm{Ker}\,\varkappa$. Ввиду $b\in N_G(S)$ имеем $\langle b,S\rangle = S\lambda\langle b\rangle$. Лемма доказана.

Зафиксируем для дальнейших рассуждений силовскую p-подгруппу S так, что $b\in N_G(S)$ и $\langle b,S\rangle = S\lambda\langle b\rangle$.

Лемма 3.6. Выполняются равенства

$$S=\{[b,s]\mid s\in S\}\cdot C_S(b), \tag{3.I4}$$

$$G=S\lambda\langle b\rangle. \tag{3.I5}$$

Доказательство. Покажем сначала равенство (3.I4). Пусть $s\in S$. Тогда $b^{-1}s^{-1}bs\in S$ и $[b,s]=b^{-1}s^{-1}bs\in S$. Предположим, что $[b,s]$ и $[b,s']$ принадлежат одному и тому же смежному классу по $C_S(b)$ $(s,s'\in S)$. Тогда $[b,s]=[b,s']c$ для некоторого $c\in C_S(b)\subset Z(G)$ и $s^{-1}bs=b\cdot[b,s]=b[b,s']c=s'^{-1}bs'\cdot c$. Порядки элементов $s^{-1}bs$ и $s'^{-1}bs'$ равны числу q_ν^ν, а c – p-элемент из центра. Поэтому $c=1$ и $[b,s]=[b,s']$. Следовательно, различные элементы вида $[b,s]$, $s\in S$, принадлежат различным смежным классам по $C_S(b)$. Но $[b,s]=[b,s']$ тогда и только тогда, когда $s^{-1}bs=s'^{-1}bs'$, т.е. когда s и s' принадлежат одному смежному классу по $C_S(b)$. Поэтому количество различных элементов вида $[b,s]$ равно количеству смежных классов группы S по $C_S(b)$. Следовательно, имеет место равенство (3.I4).

Теперь выведем соотношение

$$S\lambda\langle b\rangle \triangle G. \tag{3.I6}$$

В силу (3.I2) и (3.I4) достаточно доказать, что $g^{-1}bg$, $g^{-1}[b,s]g\in\langle b,S\rangle$ для каждых $g\in G$, $s\in S$. Поскольку $b\varkappa=b$ (ведь $\mathrm{Im}\,\varkappa=\langle b\rangle$) и согласно равенствам (3.8) и (3.II) $x\hat{g}=x\hat{s}_1$ для некоторого $s_1\in S$, то

$$g^{-1}bg=b\hat{g}=b(x\hat{g})=b(x\hat{s}_1)=bs_1=s_1^{-1}bs_1\in\langle b,S\rangle.$$

В силу (3.I2) и (3.I4) $xs\hat{g}=x\hat{s}_2$ для некоторого $s_2\in S$. Тогда

$$b(xs\hat{g})=b(x\hat{s}_2),\quad bs_2=b(sg),$$

$$s_2^{-1}bs_2=(sg)^{-1}b(sg)=g^{-1}s^{-1}bs\cdot g=g^{-1}b[b,s]g=$$

$$=g^{-1}bg\cdot g^{-1}[b,s]g,$$

т.е. $g^{-1}[b,s]g\in\langle b,S\rangle$, ибо уже доказано, что $g^{-1}bg\in\langle b,S\rangle$. Соотношение (3.I6) установлено.

Покажем равенство (3.I5). Так как имеет место соотношение (3.I6), то по теореме Шура (см. [I], стр. 378) существует такая $\{p,q\}'$-подгруппа C группы G, что $G=(S\lambda\langle b\rangle)\lambda C$. Обо-

значим через y проекцию группы G на C. Явно $y \neq 1$. Если $y \neq 0$, то $y \in J_0(G)$ и ввиду (3.8) y имеет вид $y = x\hat{g}$, т.е. $Jmy = C = Jm(x\hat{g}) \simeq Jmx \simeq C_{q^\nu}$, что невозможно, ибо $C - \{p,q\}'$-группа. Следовательно, $y = 0$ и $G = Ker\, y = S\lambda\langle b\rangle$. Лемма доказана.

Лемма 3.7. Пусть конечная группа P удовлетворяет условиям: а) P некоммутативна; б) $P = H\lambda\langle b\rangle$, где H - p-подгруппа; в) $\langle b\rangle \simeq C_{q^\nu}$, где p,q - различные простые числа $(\nu \geqslant 1)$; г) $b^q \in Z(P)$; д) H - абелева группа; е) если $h \in H$, $h \neq 1$, то $P = \langle h, b\rangle$. Тогда P - группа Миллера-Морено с параметрами p, q и ν. Если же вместо а), д), е) выполняется а') P нильпотентна; д') $C_H(b) \subset Z(P)$; е') если $h \in H \setminus C_H(b)$, то $P = \langle h, b\rangle$, то P является группой Шмидта с параметрами p, q и ν.

Доказательство. Пусть выполнены предположения леммы. Так как $\langle b\rangle$ - силовская q-подгруппа группы P, все силовские q-подгруппы сопряжены между собой и любая q-подгруппа содержится в некоторой силовской q-подгруппе, то все q-элементы группы P имеют вид $g^{-1}b^i g$.

Пусть N - собственная подгруппа группы P. В первом случае (выполняются условия а)-е)) нам надо доказать, что N - абелева, во втором случае (выполняются условия а'), б)-г), д'), е')) надо проверить, что N - нильпотентна. Если N не содержит q-элементов порядка q^ν, то все q-элементы группы N имеют по условию г) вид $g^{-1}b^{iq}g = b^{iq}$ и поэтому $N \subset \langle H, b^q\rangle = H \times \langle b^q\rangle$. Тогда в первом случае группа N коммутативна, ибо по условию д) группа H коммутативна. Во втором случае группа N нильпотентна как подгруппа нильпотентной группы $H \times \langle b^q\rangle$.

Предположим теперь, что группа N содержит элемент порядка q^ν. Тогда $g^{-1}b^i g \in N$ для некоторых $i \in Z_{q^\nu}$, число i не делится на q, $g \in P$. Поэтому $g^{-1}bg \in N$, $b \in gNg^{-1}$.

Рассмотрим сначала первый случай. Тогда $H \cap gNg^{-1} = \langle 1\rangle$. Действительно, если $H \cap gNg^{-1} \neq \langle 1\rangle$, то $gNg^{-1} = P$, ибо группа P по условию е) порождается элементом b и любым неединичным элементом из H. Но gNg^{-1} как и N - собственная подгруппа группы P. Следовательно, $H \cap gNg^{-1} = \langle 1\rangle$. Отсюда следует ввиду $b \in gNg^{-1}$ и равенства $P = H\lambda\langle b\rangle$, что $gNg^{-1} = \langle b\rangle$, т.е. группа $N = \langle g^{-1}bg\rangle$ коммутативна. Следовательно, в первом случае каждая собственная подгруппа группы P коммутативна и P является группой Миллера-Морено. Ясно, что p, q, ν - параметры группы P.

Рассмотрим теперь второй случай. Было получено, что $b \in gNg^{-1}$. Так как все p-элементы группы P содержатся в H, то группа $H \cap gNg^{-1}$ является силовской p-подгруппой группы gNg^{-1}. Если $H \cap gNg^{-1} \not\subset C_H(b)$, то существует $s \in (H \cap gNg^{-1}) \setminus C_H(b)$. По условию e') $P = \langle s, b \rangle$. С другой стороны, $\langle s, b \rangle \subset gNg^{-1}$. Поэтому $P = gNg^{-1}$, $N = g^{-1}Pg = P$, что противоречит предположению $N \neq P$. Следовательно, $H \cap gNg^{-1} \subset C_H(b)$. Отсюда в силу $C_H(b) \subset Z(P)$ имеем $gNg^{-1} = (H \cap gNg^{-1}) \times \langle b \rangle$, т.е. группа gNg^{-1} является прямым произведением своих силовских подгрупп и поэтому она, а также группа N, нильпотентна. Следовательно, каждая собственная подгруппа N группы P нильпотентна и поэтому группа P является группой Шмидта. Ясно, что p, q, ν – параметры группы P. Лемма доказана.

Лемма 3.8. Обозначим $\bar{G} = G/C_S(b)$, $\bar{S} = S/C_S(b)$, $\bar{b} = b \cdot C_S(b)$. Тогда:

1) $\bar{S}$ – неединичная элементарная абелева p-группа;

2) $\bar{G} = \bar{S} \lambda \langle \bar{b} \rangle$;

3) G не имеет нормальных делителей индекса p;

4) элемент $\bar{b}$ задает на $\bar{S}$ нетождественный внутренний автоморфизм;

5) группа $\bar{G}$ порождается элементом $\bar{b}$ и любым неединичным элементом из $\bar{S}$.

Доказательство. По лемме 3.4 $C_S(b) \subset Z(G)$. Поэтому $S \cap Z(G) = C_S(b)$ и $\hat{S} \simeq S/(S \cap Z(G)) = \bar{S}$. По лемме 4 из [5] $\hat{S} \subset V_G(x)$. В силу условия 8) группа $\hat{S} \simeq \bar{S}$ является элементарной абелевой p-группой. Предположим, что $S = C_S(b)$. Тогда в силу равенства (3.I5) $G = S \times \langle b \rangle$. Но тогда группа G коммутативна, $x \cdot \hat{g} = x$ для каждого $g \in G$ и согласно равенству (3.8) $J_0(G) = \{x\}$, что противоречит условию 4). Следовательно, $\bar{S} \neq \langle 1 \rangle$. Из равенства (3.I5) следует также равенство $\bar{G} = \bar{S} \lambda \langle \bar{b} \rangle$. Этим доказаны утверждения I) и 2) леммы.

Докажем утверждение 3). Пусть M – нормальный делитель индекса p в группе G. Тогда $G/M \simeq C_p$. Пусть gM – образующий элемент группы G/M и h – произвольный элемент порядка p группы G. Рассмотрим эндоморфизм τ группы G, являющийся произведением естественного гомоморфизма $G \to G/M$ и изоморфизма $G/M = \langle gM \rangle \to \langle h \rangle$, где $gM \mapsto h$. По построению $\tau \neq 0$, τ – собственный эндоморфизм группы G и $\mathrm{Im}\tau \simeq C_p$. В силу условия 6) $\tau \in K_G(y)$ для некоторого $y \in J_0(G)$. Отсюда $\tau y = y\tau = \tau$, т.е. $\mathrm{Im}\tau \subset \mathrm{Im}y$. Ввиду (3.8) $y = x\hat{g}_1$ для некоторого $g_1 \in G$. Поэтому

$$C_p \simeq Jmt \subset Jmy = Jm(x\hat{g}_1) = (Jmx)\hat{g}_1 \simeq$$

$$\simeq Jmx \simeq C_{q^\nu},$$

что невозможно. Следовательно, группа G не содержит нормальных делителей индекса p и утверждение 3) получено.

Для доказательства утверждения 4) заметим, что по лемме 3.6 будет $S \Delta G$ и подгруппа $C_S(b)$ инвариантна относительно автоморфизма $\hat{b}$. Поэтому можно рассматривать внутренний автоморфизм $\mu = \hat{b}|\bar{S}$ группы $\bar{S}$, порожденный элементом $\bar{b}$:

$$\bar{s}\mu = \overline{b^{-1}sb} = \bar{s}\hat{\bar{b}}, \quad s \in S.$$

Докажем, что порядок автоморфизма μ равен q_ν. Так как $\bar{s}\mu^{q_\nu} = \overline{b^{-q_\nu}sb^{q_\nu}} = \bar{s}$, то $\mu^{q_\nu} = 1$ и порядок автоморфизма μ равен q_ν или 1. Если $\mu = 1$, то ввиду утверждения 2) группа $\bar{G}$ разлагалась бы в прямое произведение $\bar{G} = \bar{S} \times \langle \bar{b} \rangle$. Тогда существовал бы такой $M \lhd G$, что $G/M \simeq C_p$ (ведь по утверждению 1) $\bar{S}$ – неединичная элементарная абелева p-группа). Это противоречит утверждению 3). Следовательно, порядок автоморфизма μ равен q_ν.

Докажем утверждение 5). Учтем, что группу $\bar{S}$ можно разложить в прямое произведение

$$\bar{S} = \bar{S}_1 \times \ldots \times \bar{S}_k,$$

где $\bar{S}_1, \ldots, \bar{S}_k$ – минимальные неединичные подгруппы, инвариантные относительно автоморфизма μ (см. [9], гл. 3, теорема 3.2). Теперь установим, что для подходящего сомножителя (например, $\bar{S}_1$) группа $\langle \bar{S}_1, \bar{b} \rangle$ удовлетворяет условиям леммы 3.7. По утверждению 2) имеем $\bar{G} = \bar{S}\lambda\langle \bar{b} \rangle$, откуда $\langle \bar{S}_i, \bar{b} \rangle = \bar{S}_i \lambda \langle \bar{b} \rangle$. Так как по утверждению 4) автоморфизм μ не является тождественным $\bar{S}$, то существует хоть одна группа $\bar{S}_i$, на которой μ не действует тождественно. Без ограничения общности можно предполагать, что μ не действует тождественно на $\bar{S}_1$. Так как μ – внутренний автоморфизм группы $\bar{S}_1 \lambda \langle \bar{b} \rangle$, то эта группа некоммутативна, т.е. выполнено условие а) леммы 3.7. Свойства б) и д) леммы 3.7 имеют место, ибо $\bar{S}_1$ является элементарной абелевой p-группой как подгруппа группы $\bar{S}$. Поскольку по утверждению 2) $\bar{G} = \bar{S}\lambda\langle \bar{b} \rangle$, то $\bar{S} \cap \langle \bar{b} \rangle = \langle 1 \rangle$ и поэтому порядок элемента $\bar{b}$ равен q_ν^ν, т.е. имеет место свойство в). По лемме 3.4 $\bar{b}^{q_\nu} \in Z(\bar{G})$. Группа $\bar{S}_1 \lambda \langle \bar{b} \rangle$ порождается элементом $\bar{b}$ и любым неединичным элементом из $\bar{S}_1$. Именно, если $\bar{s} \in \bar{S}_1$ и $\bar{s} \neq 1$,

то подгруппа, порожденная элементами $\bar{s}$ и $\bar{b}$, т.е. инвариантная относительно μ, должна совпасть с $\bar{S}_1$, иначе это противоречило бы минимальности $\bar{S}_1$. Следовательно, выполнены свойства а)-е) леммы 3.7. По этой лемме $\bar{S}_1 \lambda \langle \bar{b} \rangle$ – группа Миллера-Морено с параметрами p, q и ν. По строению групп Миллера-Морено (см. работу [IO]) известно, что порядок группы $\bar{S}_1$ равен p^u.

Покажем теперь, что $\bar{G} = \bar{S}_1 \lambda \langle \bar{b} \rangle$. Из этого будет вытекать утверждение 5) для $\bar{G}$. В силу $p^u = |\bar{S}_1| = |S_1 : C_S(b)| = [S_1 : C_S(b)]$, $S_1 \subset S$ и равенства (3.IO) имеем $S = S_1$, $\bar{S} = \bar{S}_1$, $\kappa = 1$. Теперь из утверждения 2) получаем, что $\bar{G} = \bar{S} \lambda \langle \bar{b} \rangle = \bar{S}_1 \lambda \langle \bar{b} \rangle$ и группа $\bar{G}$ порождается элементом $\bar{b}$ и любым неединичным элементом $\bar{b}$ из $\bar{S}$. Лемма доказана.

Доказательство достаточности для теоремы 3.I. Пусть для конечной группы G существует $x \in J_0(G)$, удовлетворяющий условиям I)-8) теоремы 3.I. Покажем, что группа G является группой Шмидта с параметрами p, q и ν. Для этого установим, что G удовлетворяет условиям леммы 3.7. При этом сохраняем обозначения лемм 3.3-3.8.

Разложение $G = S \lambda \langle b \rangle$ имеет место по лемме 3.6, условия в), г) и д') выполняются соответственно согласно леммам 3.3 и 3.4. Проверим, что группа G ненильпотентна. Действительно, в противном случае она разлагалась бы в прямое произведение своих силовских подгрупп, откуда $\bar{G} = \bar{S} \times \langle \bar{b} \rangle$, что невозможно, так как по утверждению 4) леммы 3.8 $\mu = \hat{\bar{b}} | \bar{S} \neq 1$. Следовательно, имеет место условие а').

Для проверки условия е') допустим, что $s \in S \setminus C_S(b)$. Тогда $\bar{s}$ – неединичный элемент группы $\bar{S}$ и поэтому по утверждению 5) леммы 3.8 $\bar{G} = \langle \bar{s}, \bar{b} \rangle$, $G = \langle b, s, C_S(b) \rangle$. Отсюда следует, что подгруппа $H = \langle b, s \rangle$ нормальна в G, ибо $G_S(b) \subset Z(G)$. Каждый элемент группы $G/H = \tilde{G}$ имеет вид aH, где $a \in S$. Поэтому каждый элемент группы $\tilde{G}$ является p-элементом. Следовательно, $\tilde{G}$ – p-группа. Если $H \neq G$, то группа G/H являлась бы неединичной p-группой и поэтому существовал бы такой $M \lhd G$, что $G/M \simeq C_p$. Это противоречит утверждению 3) леммы 3.8. Следовательно, $G = H$ и группа G порождается элементом b и любым неединичным элементом из $S \setminus C_S(b)$.

Для группы G выполнены теперь предположения а'), б)-г), д') и е') леммы 3.7 Поэтому группа G является группой Шмидта с параметрами p, q и ν. Теорема доказана.

Теорема 3.9. Если G – группа Шмидта с параметрами p, q и ν и полугруппы всех эндоморфизмов групп G и H изоморфны, то H – группа Шмидта с параметрами p, q и ν.

Доказательство. Пусть выполнены предположения теоремы. Из изоморфизма $EndG \simeq EndH$ и конечности полугруппы $EndG$ следует конечность группы H (см. [8], теорема 2). Пусть $T: EndG \to EndH$ – заданный изоморфизм. По теореме 3.I существует $x \in J_0(G)$, удовлетворяющий условиям I)-8) теоремы 3.I. В силу изоморфизма T идемпотент $xT \in J_0(H)$ удовлетворяет условиям, аналогичным условиям теоремы 3.I. Поэтому группа H является группой Шмидта с параметрами p, q и ν. Теорема доказана.

Отметим, что в теореме 3.9 нельзя утверждать, что группы G и H изоморфны (см. [2], теорема 3.7).

Литература

I. Курош А.Г. Теория групп. М., I967.

2. Пуусемп П.А. Полугруппы эндоморфизмов групп Шмидта. – Тр. Таллинск. политехн. ин-та, I983, № 554, с.I55-I64.

3. Пуусемп П.А. Об автоморфизмах групп Шмидта.– Тр. Таллинск. политехн. ин-та, I983, № 554, с. I65-I68.

4. Пуусемп П. Идемпотенты полугрупп эндоморфизмов групп. – Уч. зап. Тартуского гос. ун-та, I975, № 366, с. 76-I04.

5. Пуусемп П. Полугруппа эндоморфизмов полупрямого произведения двух циклических p-групп. – Уч. зап. Тартуского гос. ун-та, I976, № 390, с. I04-I33.

6. Пуусемп П. Полугруппы эндоморфизмов обобщенных групп кватернионов. – Уч. зап. Тартуского гос. ун-та, I976, № 390, с. 84-I03.

7. Холл М. Теория групп. М., I962

8. Alperin J.L. Groups with finitely many automorphisms. – Pacif. J. Math., 1962, 12, N 1, p. 1-5.

9. G o r e n s t e i n D. Finite groups. New York, 1968.

10. R e d e i L. Das "schiefe Produkt" in der Gruppentheorie ... - Comm. Math. Helv., 1947, N 20, S. 225-264.

11. R e d e i L. Die endlichen einstufig nichtnilpotenten Gruppen. - Publ. Math., 1956, N 4, S. 303-324.

P. Puusemp

An Abstract Characteristic of the Schmidt Groups by Means of Their Endomorphism Semigroups

S u m m a r y

A finite non-nilpotent group is said to be a Schmidt group if all its proper subgroups are nilpotent. The construction of the Schmidt groups is well-known (L. Redei [10, 11]). In the papers [2, 3] the semigroups of all endomorphisms and the groups of all automorphisms of the Schmidt groups are characterized. In this paper the following proposition for the Schmidt groups is proved: if the semigroup of all endomorphisms of an arbitrary group G is isomorphic to the semigroup of all endomorphisms of some Schmidt group then the group G is always a Schmidt group. This result shows that the class of all Schmidt groups is determined by their semigroups of endomorphisms.

№ 568
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 511

П.П. Таммела

ОБ ОДНОМ СЕМЕЙСТВЕ ОБЛАСТЕЙ ПРИВЕДЕНИЯ ВЕНКОВА ПОЛОЖИТЕЛЬНО ОПРЕДЕЛЕННЫХ КВАДРАТИЧНЫХ ФОРМ

I. Введение

Эта работа посвящена теории приведения положительных квадратичных форм по Б.А. Венкову [2]. С определениями и результатами теории приведения положительных квадратичных форм можно ознакомиться в работах [1-12].

Приведем сначала некоторые обозначения и определения [2]. Пусть

$$f(x)=\sum_{i,j=1}^{n}a_{ij}x_ix_j,\quad g(x)=\sum_{i,j=1}^{n}b_{ij}x_ix_j\ (a_{ij}=a_{ji},\ b_{ij}=b_{ji},\ i,j=1,\dots,n)$$

две положительно определенные квадратичные формы с вещественными коэффициентами положим

$$(f,g)=\sum_{i,j=1}^{n}a_{ij}b_{ij}=a_{11}b_{11}+\dots+a_{nn}b_{nn}+2a_{12}b_{12}+\dots+2a_{n-1n}b_{n-1n}. \quad (1)$$

Выражение (f,g) называется иногда скалярным или внутренним произведением форм $f(x)$ и $g(x)$, а также полуинвариантом Вороного [4]. При

$$g(x)=\sum_{i=1}^{k}\mu_i(\lambda_{i1}x_1+\dots+\lambda_{in}x_n)^2$$

имеем

$$(f,g)=\sum_{i=1}^{k}\mu_i f(\lambda_{i1},\dots,\lambda_{in}). \quad (2)$$

Форма $F(x)=\sum_{i,j=1}^{n}A_{ij}x_ix_j=\overline{f}(x)$ называется взаимной к форме f, если коэффициенты A_{ij} являются алгебраическими дополнениями элементов a_{ij} в определителе матрицы (a_{ij}) формы f. Через fS обозначается квадратичная форма, получающаяся из формы f целой унимодулярной подстановкой S.

ОПРЕДЕЛЕНИЕ. Пусть φ данная положительно определенная квадратическая форма, $\bar{\varphi}$ - взаимная с нею форма. Пусть S пробегает множество всех целых унимодулярных матриц. Обозначим через $V(\varphi)$ множество всех положительных квадратичных форм f, удовлетворяющих всем неравенствам

$$(f,\bar{\varphi}) \leq (f,\bar{\varphi}S). \tag{3}$$

Форму φ называем базисной формой области приведения Венкова $V(\varphi)$, а форма f называется φ-приведенной или формой, приведенной по Венкову относительно базисной формы φ.

Ненулевой целый вектор $\vec{l}=(l_1,l_2,\dots,l_n)$ называется вектором смежности формы f, если для всех векторов $\vec{l}_\varphi$ сравнимых с вектором $\vec{l}$ по mod2 (т.е. все соответствующие координаты сравнимы по mod2), выполняется неравенство

$$f(\vec{l}) \leq f(\vec{l}_1).$$

Мы обозначаем через e_i вектор, у которого все координаты, кроме i-той координаты, равны нулю, а i-тая координата равна единице.

Основная цель этой работы дать доказательство и обобщение теоремы 5 из работы [9].

ТЕОРЕМА I. Все области приведения Венкова $V(\varphi_t)$, где

$$\bar{\varphi}_t = x_1^2+x_2^2+x_3^2+2t(x_1x_2+x_1x_3+x_2x_3),\ 0<t<1 \tag{4}$$

определяются неравенствами:

$$a_{ij}+a_{ik} \leq 0 \tag{5}$$

$$a_{ii}+a_{ij}+a_{ik} \geq 0 \tag{6}$$

$$a_{ii}-a_{ij}+a_{ik} \geq 0 \tag{7}$$

$$(2t-1)(a_{ii}+a_{jj}+2a_{ij}+2a_{ik}+2a_{jk}) \leq 0 \tag{8}$$

$$(2t-1)(a_{ii}+2a_{ij})+2ta_{ik} \leq 0 \tag{9}$$

$$(2t-1)(a_{ii}+2a_{ij})+2ta_{ik}+4ta_{jk} \leq 0 \tag{10}$$

$$(2t-1)(a_{ii}+a_{jj}+2a_{ij}+2a_{jk})+2ta_{ik} \leq 0 \tag{11}$$

$$2(t-1)(a_{ii}+a_{ik})+(2t-1)(a_{jj}+2a_{jk})+2(3t-2)a_{ij} \leq 0 \tag{12}$$

$$2(t-1)(a_{jj}+a_{kk}+2a_{jk})+2a_{ik}+2ta_{ij} \leq 0 \tag{13}$$

$$(2t-1)(a_{ii}+a_{kk}+2a_{jk})+2ta_{ij}+2a_{ik}\le 0 \qquad (14)$$

$$(2t-1)(a_{jj}+2a_{jk})+2(t-1)a_{ii}+2(3t-2)a_{ij}+2(3t-1)a_{ik}\le 0 \qquad (15)$$

здесь i,j,k пробегают все перестановки индексов I, 2, 3.

Описание матриц S, соответствующих неравенствам (5) - (I5) как неравенствам (3), будет дано при доказательстве теоремы I в § 2. В § 3 мы выводим указанный результат работы [9].

§ 2. Доказательство теоремы I.

В этом параграфе через $S=(\vec{S}_1,\vec{S}_2,\vec{S}_3)$ мы обозначаем целую унимодулярную матрицу со столбцами $\vec{S}_1,\vec{S}_2,\vec{S}_3$ и через $\vec{S}_0$ обозначаем вектор $\vec{S}_1+\vec{S}_2+\vec{S}_3$. Через $f(x\,y)$ мы обозначаем билинейную форму, связанную с квадратичной формой $f(x)$. При этом $f(e_i;e_j)=a_{ij}$, $i,j=1,2,3$.

Перед доказательством теоремы докажем две леммы.

ЛЕММА I. Пусть положительно определенная квадратичная форма f удовлетворяет строго неравенствам (5) - (7), тогда векторы смежности формы f имеют координаты, не превосходящие по абсолютной величине I.

ДОКАЗАТЕЛЬСТВО. Из неравенств (5) - (7) и из положительной определенности формы f получаем еще неравенства

$$a_{ii}\pm a_{ij}>0,\quad a_{ii}-2a_{ij}>0,\quad a_{ii}-a_{ij}-a_{ik}>0, \qquad (16)$$

где i,j,k пробегают все перестановки индексов I, 2, 3.

Рассмотрим значения $f(l_1,l_2,l_3)$ формы f, удовлетворяющей условиям леммы. Без потери общности мы можем считать, что

$$0\le|l_1|\le|l_2|\le|l_3| \qquad (17)$$

и при этом можем рассматривать только случаи, когда $l_3\ge 1$.

Рассмотрим следующие три случая:

I. Если

$$l_3>1,\quad l_3>|l_2|, \qquad (18)$$

тогда в силу (6) - (7) и (I6) имеем

$$f(l_1,l_2,l_3)-f(l_1,l_2,l_3-2)=$$

$$= 4|l_1|[f(e_1;e_3)\operatorname{sign} l_1 + f(e_2;e_3)\operatorname{sign} l_2 + f(e_3)] +$$
$$+ 4(|l_2| - |l_1|)[f(e_2;e_3)\operatorname{sign} l_2 + f(e_3)] +$$
$$+ 4(l_3 - 1 - |l_2|) f(e_3) > 0. \qquad (19)$$

2. Если

$$|l_3| = |l_2| > |l_1|, \quad l_3 > 1, \qquad (20)$$

тогда в силу (6), (7) и (16) имеем

$$f(l_1,l_2,l_3) - f(l_1,l_2-2\operatorname{sign} l_2,\ l_3-2\operatorname{sign} l_3) =$$
$$+4|l_1|[f(e_1;e_2)\operatorname{sign} l_1\operatorname{sign} l_2 + f(e_2) + f(e_2;e_3)\operatorname{sign} l_2\operatorname{sign} l_3] +$$
$$+4|l_1|[f(e_1;e_3)\operatorname{sign} l_1\operatorname{sign} l_3 + f(e_2;e_3)\operatorname{sign} l_2 + \operatorname{sign} l_3 + f(e_3)] +$$
$$+4(|l_2| - 1 - |l_1|) f(e_2\operatorname{sign} l_2 + e_3\operatorname{sign} l_3) > 0. \qquad (21)$$

3. Если

$$|l_1| = |l_2| = |l_3| > 1, \qquad (22)$$

тогда имеем

$$f(l_1, l_2, l_3) = l_1^2 f(\operatorname{sign} l_1, \operatorname{sign} l_2, \operatorname{sign} l_3). \qquad (23)$$

Условия (18), (20) и (22) дают все случаи, когда максимальная абсолютная величина координат векторов $\underline{l} = (l_1, l_2, l_3)$ больше I, а в силу неравенств (19), (21) и (23) получаем, что вектор $\vec{l}$ в этих случаях не является вектором смежности формы f. Следовательно, все координаты векторов смежности по абсолютной величине не превосходят единицу.

Лемма доказана.

По лемме I у положительно определенной формы, удовлетворяющей строго неравенствам (5) - (7), имеются следующие векторы смежности $\pm e_i, \pm(e_i \pm e_j), \pm(e_1 + e_2 + e_3)$, где $i, j = 1, 2, 3, i \neq j$, причем одновременно в силу (5) не более одной пары векторов смежности $\pm m_i$ могут иметь вид $\pm(e_i - e_j)$.

ЛЕММА 2. В каждом классе сравнения по mod 2 целых унимодулярных матриц имеется матрица $S = (\vec{S}_1, \vec{S}_2, \vec{S}_3)$ такая, что все векторы $\vec{S}_1, \vec{S}_2, \vec{S}_3$ и $\vec{S}_0 = \vec{S}_1 + \vec{S}_2 + \vec{S}_3$ являются векторами смежности формы f, удовлетворяющей строго неравенствам (5), (6) и (7).

ДОКАЗАТЕЛЬСТВО. Заметим, что если матрица $S=(\vec{S}_1,\vec{S}_2,\vec{S}_3)$ удовлетворяет условиям леммы 2, тогда лемме 2 удовлетворяют и матрицы $(\vec{S}_i,\vec{S}_j,\vec{S}_k)$, где i,j,k любая перестановка индексов I, 2, 3.

Легко показать, что любая матрица сравнима по $\bmod 2$, с точностью до перестановки столбцов, с одной из следующих матриц:

$$\begin{gathered}
\pm(e_i,e_j,e_k),\ \pm(e_i,e_j,-(e_1+e_2+e_3)),\ \pm(e_i,-e_j,e_j+e_k),\\
\pm(e_i,-(e_i+e_k),e_j+e_k),\ \pm(e_i,e_j,-e_j-e_k),\ \pm(e_i,e_j,-e_i+e_k),\\
\pm(e_i,-e_i+e_k,-e_j-e_k),\ \pm(-e_i+e_k,e_j,-e_j-e_k),\ \pm(e_i,-e_i+e_j,e_i+e_j+e_k),\\
\pm(-e_i,e_i+e_j,e_i+e_k),\ \pm(e_i+e_j,e_i+e_k,-e_i-e_j-e_k),\ \pm(-e_i,-e_i-e_j,e_i+e_j+e_k),\\
\pm(e_i,e_i+e_j,-e_i+e_k),\ \pm(-e_i,e_i-e_k,e_i+e_j+e_k),\ \pm(-e_i-e_j,e_i-e_k,e_i+e_j+e_k),
\end{gathered}\tag{24}$$

здесь i,j,k пробегают все перестановки индексов I, 2, 3.

Все такие матрицы удовлетворяют условиям леммы.

Лемма доказана.

ДОКАЗАТЕЛЬСТВО ТЕОРЕМЫ I. Так как

$$(f,\bar{\varphi}_t S')=(fS,\bar{\varphi}_t),\tag{25}$$

где S' - транспонированная матрица матрицы S, то в силу [2] и того, что

$$\bar{\varphi}_t=(1-t)(x_1^2+x_2^2+x_3^2)+t(x_1+x_2+x_3)^2\tag{26}$$

имеем

$$(fS,\bar{\varphi}_t)=(1-t)(f(\vec{S}_1)+f(\vec{S}_2)+f(\vec{S}_3))+tf(\vec{S}_1+\vec{S}_2+\vec{S}_3),\tag{27}$$

где $\vec{S}_1,\vec{S}_2,\vec{S}_3$ столбцы матрицы S.

В силу леммы 2 имеем, что для формы f, удовлетворяющей строго неравенствам (5) - (7), минимум $\min\limits_{S\in GL(n,Z)}(fS,\bar{\varphi}_t)$ может достигаться только на матрицах $S=(\vec{S}_1,\vec{S}_2,\vec{S}_3)$, где столбцы $\vec{S}_1$, $\vec{S}_2,\vec{S}_3$ и их сумма $\vec{S}_1+\vec{S}_2+\vec{S}_3$ являются векторами смежности.

В силу определения области приведения Венкова $V(\varphi_t)$ единичная матрица является всегда представлением минимума выражения

$$(f,\bar{\varphi}_t S')=(fS,\bar{\varphi}_t)\tag{28}$$

для f приведенных по Венкову $f\in V(\varphi_t)$ (см. неравенство (3)). Неравенства (5) соответствуют

$$(f, \bar{\varphi}_t) \le (fS, \bar{\varphi}_t), \qquad (29)$$

где $S = (-e_i, e_j, e_k)$. Неравенства (6) соответствуют неравенствам (29), где матрицы $S = (-e_i, e_i+e_j, e_i+e_k)$, а неравенства (7) соответствуют неравенствам (29), где матрицы $S = (e_i, e_i+e_j, -e_i+e_k)$.

В силу определения векторов смежности, если $S = (\vec{S}_1, \vec{S}_2, \vec{S}_3)$ не состоит из векторов смежности или вектор $\vec{S}_1 + \vec{S}_2 + \vec{S}_3$ не является вектором смежности, то неравенство (29) является следствием неравенств (5) - (7) и неравенства (29), где матрица S принадлежит списку (24).

Из неравенств (9), (II) и (I6) получаем

$$2tf(e_i; e_j) + (2t-1)[f(e_j) + 2f(e_j; e_k)] + 2(t-1)[f(e_i) + f(e_i; e_k)] \le 0, \qquad (30)$$

$$2(t-1)[f(e_i) + f(e_i; e_j)] + (2t-1)[f(e_j) + f(e_k) + 2f(e_j; e_k) + 2f(e_i; e_j)] + 2tf(e_i; e_k) \le 0. \qquad (31)$$

которые соответствуют матрицам

$$\pm(e_i, -e_i-e_j, e_i+e_j+e_k), \quad \pm(-e_i-e_j, e_i-e_k, e_i+e_j+e_k).$$

Из неравенств (5) и (9) получаем

$$(2t-1) f(e_i) + 2f(e_i; e_k) + 2tf(e_i; e_j) \le 0, \qquad (32)$$

которые соответствуют матрицам $\pm(e_i, e_j, -e_i+e_k)$.

Из неравенств (6) и (8) получаем неравенства

$$2(t-1) f(e_i) + (2t-1)[f(e_j) + f(e_k) + 2f(e_j, e_k)] + (6t-4)(f(e_i, e_k) + f(e_i, e_j)) \le 0, \qquad (33)$$

которые соответствуют матрицам $\pm(e_i+e_j, e_i+e_k, -(e_i+e_j+e_k))$.

Теорема I доказана.

§ 3. Случай $0 < t < \frac{1}{2}$ теоремы I.

ТЕОРЕМА 2. Область приведения $V(\varphi_t)$ при $0 < t < \frac{1}{2}$ определяется неравенствами (5), (6), (8) - (I2) теоремы I.

ДОКАЗАТЕЛЬСТВО. Теорема следует из теоремы I, так как неравенства (7), (I3) - (I5) являются следствиями других неравенств теоремы 2 при $0 < t < \frac{1}{2}$.

Заметим, что при $0 < t < \frac{1}{2}$, из неравенства (9) при $f(e_i; e_k) > 0$ получаем

$$f(e_i) + 2f(e_i; e_j) \ge 0. \qquad (34)$$

Из неравенств (5) и (34) следует неравенство (7). Из неравенств (10) и (16) следует неравенство (13). Из неравенств (5), (9) и (16) следует неравенство (14). Из неравенств (5), (8) и (34) следует неравенство (15).

Независимость условий теоремы 2 следует из теоремы Минковского-Фаркаша [10, лемма 2.4].

Теорема 2 доказана.

Литература

1. Барановский Е.П. Область приведения по Зеллингу положительных квадратичных форм от пяти переменных. - Труды МИАН СССР, 1980, т. 152, с. 5-33.

2. Венков Б.А. О приведении положительных квадратичных форм. - Изв. АН СССР, сер. матем., 1940, т. 4, № 1, с. 37-52. (см. также Венков Б.А. Избранные труды. Л., 1981, с. 188-200).

3. Делоне Б.Н. Геометрия положительных квадратичных форм. Успехи матем. наук, вып. 3, 1937, с. 16-62, вып. 4, 1938, с. 103-164.

4. Малышев А.В. Квадратичных форм приведение. - Математическая энциклопедия, М., 1979, т. 2, стб. 788-791.

5. Рышков С.С. О приведении положительных квадратичных форм от n переменных по Эрмиту, по Минковскому и по Венкову. - Докл. АН СССР, 1972, т. 207, № 5, с. 1054-1056.

6. Рышков С.С. О приведении положительных квадратичных форм по Венкову. - Учен. зап. Ивановского гос.ун-та, 1974, т. 89, с. 5-36.

7. Таммела П.П. К теории приведения положительных квадратичных форм. - Докл. АН СССР, 1973, т. 209, с. 1299-1302.

8. Таммела П.П. Область приведения Минковского для положительных квадратичных форм от семи переменных. - Зап. науч. семинаров ЛОМИ, 1977, т. 67, с. 108-143.

9. Таммела П.П. К теории приведения положительных квадратичных форм по Венкову. - Зап. науч. семинаров ЛОМИ, 1983, т. 121, с. 108-116.

IO. Черников С.Н. Линейные неравенства. М., Наука, I968. 488 с.

II. Штогрин М.И. Об областях приведения Вороного, Венкова и Минковского. - Докл. АН СССР, I972, т. 207, с. I070-I073.

12. Waerden B.L. van der. Die Reduktionstheorie der positiven quadratischen Formen. - Acta mathem., 1956, Bd. 96, S. 265-309.

P. Tammela

On the Family of Reduction Domains in the Sense of Venkov of Positive Definite Quadratic Forms

Summary

The author proves and extends one of the results of his paper "On the theory of reduction in the sense of Venkov of positive definite quadratic forms" (Zap. Naučn. Sem. Leningrad Otdel. Mat. Inst. Steklov (LOMI), v. 121, p. 108-116). A set of inequalities are presented, which determine all reduction domains $V(\varphi_t)$, where

$$\overline{\varphi}_t = x_1^2 + x_2^2 + x_3^2 + 2t\,(x_1x_2 + x_1x_3 + x_2x_3)$$

with $0 < t < 1$.

Содержание

Цена 1 руб.