ISSN 0136-3549
0320-3409

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED
645

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

TPI '87

ОБРАБОТКА ДАННЫХ,
ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ,
ВОПРОСЫ ПРОГРАММИРОВАНИЯ

645

TPI '87

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

ОБРАБОТКА ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ, ВОПРОСЫ ПРОГРАММИРОВАНИЯ

Труды экономического факультета LXII

Таллин 1987

ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Труды ТПИ № 645

ОБРАБОТКА ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ,
ВОПРОСЫ ПРОГРАММИРОВАНИЯ
Труды экономического факультета LXII
На русском языке
Отв. редактор И. Амитан. Техн. ред. А. Андриевская
Сборник утвержден коллегией Трудов ТПИ 12.06.87
Подписано к печати 11.11.87
МВ-08465
Формат 60x90/16. Печ. л. 11,0+0,5 приложение
Уч.-изд. л. 9,2. Тираж 400. Зак. № 519
Цена 1 руб. 80 коп.
Таллинский политехнический институт,
200108, Таллин, Эхитаяте тээ, 5
Ротапринт ТПИ, 200006, Таллин, ул. Коскла, 2/9

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.03

Я.Я. Тепанди

РАСКРЫТИЕ ПОНЯТИЙ ДЛЯ ПОЛЬЗОВАТЕЛЯ ПРОБЛЕМНО-ОРИЕНТИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

I. Введение

При построении проблемно-ориентированного программного обеспечения необходимо выделить существенные понятия предметной области. Эти понятия используются затем при проектировании самой системы и ее базы данных (базы знаний), при запросах к системе и т.д. [1, 2]. Обычно выделение понятий - дело человека. Развитие экспертных систем выдвинуло на первый план задачу пополнения и обобщения знаний [3, 4], причем знания эти должны предоставляться человеку в подходящем для него виде [6]. Поэтому важна и задача формирования понятий и представления их в наиболее "человеческом" виде.

Прежде, чем идти дальше, уточним, что понятие P включает в себя абстрактный объект

$$P = \langle N, D, C \rangle,$$

где N - имя понятия; D - объем (экстенсионал) понятия, то есть совокупность отраженных в понятии предметов или явлений; C - содержание понятия, то есть совокупность существенных признаков, по которым выделяются предметы и явления, входящие в данное понятие. Другими словами, D - это денотат (значение) имени N, а C - концепт (смысл), который выражает имя N. В дальнейшем вместо "понятие с именем N" будем писать просто "понятие N".

Таким образом, для формирования понятия необходимо дать ему имя, ограничить его объем и определить его содержание. Очевидно, что содержание и объем понятия могут быть связаны между собой: имея полное описание предметной области (универсума) понятия, можно, в принципе, по объему определить

содержание, и наоборот. В этом случае можно ограничиться знанием либо D , либо C, но тогда придется в определение понятия ввести описание предметной области. Если же последняя не зафиксирована или неизвестна, то неизвестна и зависимость между D и C.

Используя введенную терминологию, можно сказать,что задачей кластерного анализа (и подобных ему методов) является выделение объема понятия, т.е. класса однотипных объектов или явлений. Задача же определения содержания понятия решается, например, методами распознавания образов [7, 8]. При этом содержание понятия определяется в виде решающих правил -функций, определяемых на признаках исследуемых объектов. Эти функции подчас весьма сложны.

Человек обычно поступает по-иному, определяя одно понятие через другое. При этом возникают разные иерархии: есть-иерархия, есть-часть-иерархия и др. [II]. Интересно выяснить, каким именно образом должны быть сформулированы решающие правила, чтобы они были легко доступны и понятны человеку.

В [3] показано, что для принятия различных решений необходимо введение "слоёного пирога" описаний ситуаций. Таково же положение в области понятий. Какой же "слоёный пирог" понятий выпекли себе люди в процессе своего исторического развития? Какова связь понятий на разных ступенях различных иерархий? Как одно понятие выражается через другое?

2. Цель и метод исследования

Итак, наша конечная цель - формулировка содержания понятия (например, решающего правила, полученного в результате применения методов распознавания образов) в привычном и удобном для человека виде. Прежде, чем попытаться это сделать, мы должны выяснить, какой вид формулировки обычен и удобен для человека. Выяснение этого для определенной группы понятий - цель данной статьи. Какие же существуют понятия и где они определяются? Какова связь между понятиями и языком? Не вдаваясь в подробности (см., напр. [I2]), заметим только, что большинство (если не все) имен существительных являются именами понятий, например: скука, бесконечность, лошадь (конечно, кроме таких понятий существуют и

другие, например, "бегающий мальчишка"). Определяются же понятия почти в каждой книге, но для наших целей подходят больше всего (толковые) словари, энциклопедии, справочники и т.п., прямое назначение которых - толковать значение слов, в т.ч. имен понятий. Как уже отмечалось, мы будем вместо "понятие с именем "колпачок" писать просто "понятие "колпачок", поэтому можно сказать, что словарная статья слова "колпачок", является формулировкой содержания(определением) понятия "колпачок".

Сделаем еще допущение, что составители разных словарей в ходе долгой практики пришли к наиболее удобным и обычным для человека определениям слов (понятий). Тогда ясно, что определения понятий для пользователей проблемно-ориентированных систем целесообразно построить по аналогии с определениями понятий, данными в толковом словаре.

Итак, для ответа на поставленные в конце предыдущего параграфа вопросы были проанализированы определения 40 понятий вместе с их есть-понятиями, взятые на материале толковых словарей русского и английского языков.

Источник (толковый словарь) был выбран, исходя из двух требований: он должен был содержать определения понятий и содержать все есть-понятия. Первому требованию не соответствовали двуязычные словари, второму - энциклопедии, справочники и т.п. Из источников по русскому языку были рассмотрены словарь В. Даля и 4-томный толковый словарь под ред. А.П. Евгеньевой [9]. Последний был выбран, так как цель исследования - уяснить не историческое образование понятий (что было бы тоже интересно), а способы определения понятий у современного человека. По английскому языку был выбран Оксфордский словарь [10].

В принципе, для нужд проблемно-ориентированных систем необходимы все типы понятий, в т.ч. понятия, выражающие действия, свойства и т.п. Например, "зарезервирование" или "высокость" вполне могут оказаться важными понятиями информационной системы. В настоящей статье ограничимся, однако, только понятиями, объемы которых представляют собой классы поддающихся исчислению предметов или существ реального мира (гармонист, колпачок, маломерка, ...). Далее, под словом "понятие" будем, если не оговорено иначе, понимать именно

такое понятие-класс (другими словами - неотвлеченное предметное понятие). Предлагаемая же методика исследования может быть использована также при исследовании других видов понятий (действий, свойств и т.д.).

Из обоих языков было выбрано по 20 слов - первое значение первого понятия примерно на каждой 200-й странице словаря [9] (точнее, были выбраны понятия, начиная со с. 200, 400, 600 I-го тома; со с. I00, 300, 500, 700, 900 2-го и 3-го томов; со с. I00, 300, 500, 700, 900, I080 4-го тома) и на каждой 50-й странице словаря [I0]. Таким образом, понятия располагались равномерно по всему объему словарей и были выбраны, по существу, случайным образом. Поэтому приводимые ниже числа характеризуют пусть неточно, частоту отдельных способов определения понятий.

Приводим список рассмотренных понятий: вешалка, гармонист, драгоман, затишье, колпачок, маломерка, наконечник, нота, острица, перелог, подсказчик, предельщик, проситель, растяпа, секундант, спай, тильбюри, уроженец, чек, ящерица; attendant, branch, citron, cramp, dissertation, expo, fungus, helicopter, ion, lodging, mortarboard, overpass, post, rebel, sage, simpleton, stockade, threshold, valve, yob.

Для каждого понятия рассматривалось его есть-понятие (родовое понятие), если оно существовало, а также другие связанные понятия. Были составлены фреймовые представления [3] всех указанных понятий и проанализированы их связи. Результаты этого анализа представлены в следующих пунктах.

3. Структура определения понятия

Обычное определение понятия может включать есть-понятия, роли и примеры.

На рис. I представлено определение понятия. Оно включает понятие П и группу есть-понятий Е. Последние имеют занятые роли ЕЗ, ЕЗП и свободные роли ЕС, ЕСП. Свободные роли есть-понятий могут заполняться в понятии (группа ЕСП) или остаться незаполненными (группа ЕС). Занятые роли есть-понятий могут оставаться незатронутыми (группа ЕЗ) или переопределяться (группа ЕЗП). Роли понятия могут также

быть свободными (группа ПС) или занятыми (группы ПЗ, ЕЗП, ЕСП). Кроме того, определение понятия включает обычно типичные примеры использования (ПИ); примеры есть-понятий на рисунке не обозначены.

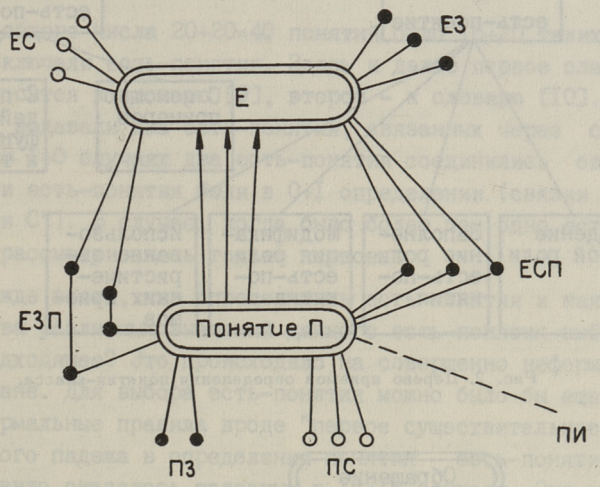


Рис. 1. Определение понятия.

Роли заполняются либо описаниями, либо перечнем примеров, либо значением по умолчанию. Описание роли - обычно самостоятельное понятие или определение понятия; такие понятия мы далее рассматривать не будем.

Дерево приемов определения понятия-класса приводится на рис. 2.

Пример. В определениях

(I) "Нота - официальное дипломатическое письменное обращение правительства одного государства к другому",

(2) "Обращение - просьба, призыв, речь, обращенные к кому-л., чему-л.",

задано понятие "нота" (рис. 3). Оно определено через есть--понятие "обращение" и пять ролей:

признаки - "официальное", "дипломатическое", "письменное";
субъект - "правительство одного государства";
объект - "другое (правительство)".

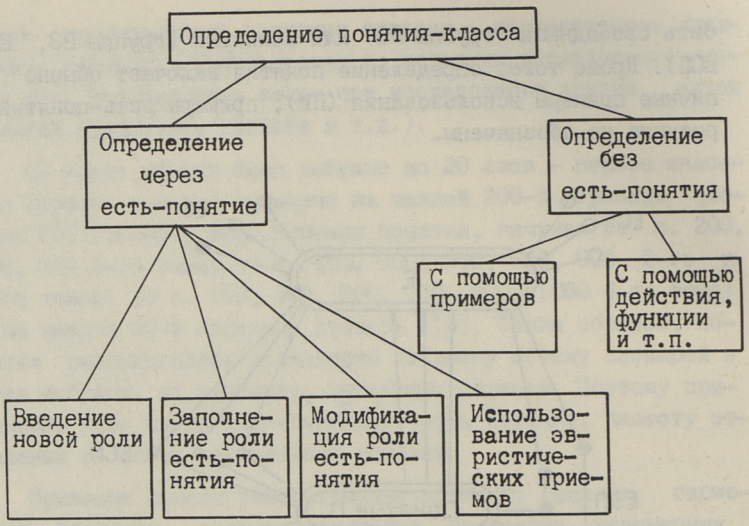


Рис. 2. Дерево приемов определения понятия-класса.

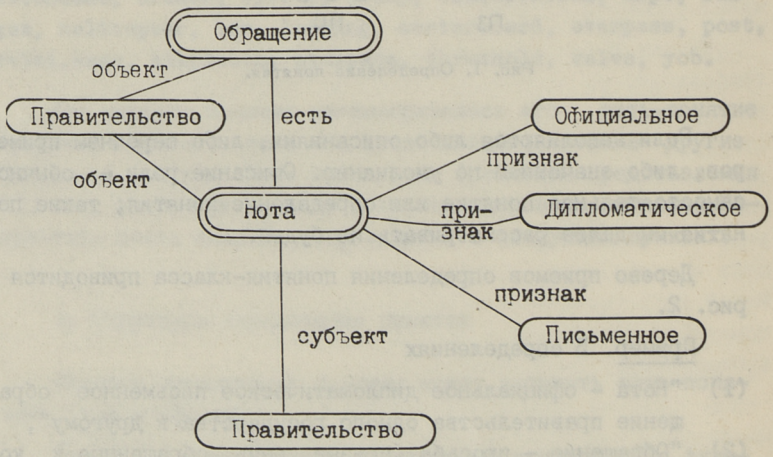


Рис. 3. Определение понятия "нота" через есть-понятие "обращение".

В есть-понятии одна роль (объект) не заполнена (" ... обращенные к кому-л., чему-л."); она заполняется в (I). Заметим, что определение есть-понятия "обращение" включает три есть-понятия, связанных, по существу, посредством связок "или".

Рассмотрим теперь подробнее возможные приемы определения понятий.

4. Определение через есть-понятие

Из общего числа 20+20=40 понятий было I6+20 таких, которые включали есть-понятия. Здесь и далее первое слагаемое относится к словарю [9], второе - к словарю [IO]. В I+3 случаях задавали два есть-понятия, связанных через связку "или". В I+0 случаях два есть-понятия соединились связкой "и". Три есть-понятия были в 0+I определении (связки "(А и В) или С"). В случае, когда было более чем одно есть-понятие, рассматривалось только первое.

Прежде всего, как определялись есть-понятия и как из множества различных значений данного есть-понятия выбиралось подходящее? Это происходило на совершенно неформальном уровне. Для выбора есть-понятия можно было бы еще задать формальные правила вроде "первое существительное именительного падежа в определении понятия - есть-понятие". Это правило оказалось полезным в I5+I8 случаях. Оно не работает, если отсутствует есть-понятие, в случае нескольких есть-понятий, а также в определениях типа "экипаж - общее название рессорных повозок ..." или "палец - один из пяти ...".

Правильный выбор среди I...9 значений есть-понятия требовал знания содержания соответствующих терминов. Кажущееся разумным правило "выбирай то значение есть-понятия, свободные роли которого заполнены в понятии" оказалось практически неприемлемым, т.к. роли обычно прибазлялись в понятие вместо того, чтобы заполнить свободные роли есть-понятия.

Рассмотрим разные способы определения через есть-понятие.

Введение новой роли в есть-понятие. Этот способ использовался наиболее часто. При этом старые роли не изменялись. Например, в определениях

(3) "Маломерка - предмет меньших, чем обычно, размеров..." и

(4) "Предмет - вещь ..., обслуживающая ту или иную потребность ..."

введена дополнительная роль "размер", характеризующая есть-понятие "предмет".

Введение новой роли - операция конкретизации есть-понятия. Наиболее часто использовалась конкретизация по свойствам (I0+I2 случаев), по размеру (5+4 случаев), по способу представления и т.д. Одно понятие конкретизируется через одно свойство, как в (3), или через несколько, как в (I).

В 7+4 случаях использовалась конкретизация по действию:

(5) "Гармонист - тот, кто играет на гармонике".

Роль "есть-часть" использовалась в 3+2 случаях, а противоположная ей - "состоит из" - в 4+4 случаях. Частота применений роли "назначение" была весьма различной для двух языков - она использовалась в I+7 случаях.

Заполнение явно указанной в есть-понятии роли. Этот способ применен, например, в определениях понятия (I) и есть-понятия (2).

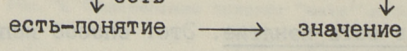
Свободные роли были явно указаны в 8+9 есть-понятиях. Они использовались в определениях 3+5 понятия. Максимальные числа свободных ролей - 2 и 3, максимальные числа используемых ролей - 2 и I.

Замена заполненной роли есть-понятия другой. Например, определения:

(6) "колтун - болезнь кожи у лошадей ..."

(7) "болезнь - нарушение нормальной жизнедеятельности организма".

Здесь роль "носитель" есть-понятия "болезнь" имеет значение "организм". Значение этой же роли в понятии ("колтун") заменено примером "кожа" понятия "организм". Это можно проиллюстрировать на следующей диаграмме:



Такая замена выполнялась в 4+5 определениях.

Использование эвристических правил формирования понятий. Например, в (I) и (2), а также в

(8) "breadth - distance or measure from side to side",

(9) "distance - measure of space, between two points, places etc".

используется обобщение следующего правила из [5]: "Если T(x,y) - функция от двух аргументов, имеющих общую область определения K, то попробуй придать обоим аргументам одинаковое значение". Обобщение можно сформулировать в виде "... попробуй разбить область определения K на интересные подмножества и выбери x,y из одного и того же подмножества". В (8) и (9) вместо "между двумя точками" подставляется "между двумя сторонами одного объекта", т.е. подмножество K состоит из точек на противоположных сторонах одного объекта.

Многие понятия, имеющие однотипные роли, преобразуются в новые понятия в результате применения этого правила, например, "любовь", "убийство", "воспитание" и др. В рассмотренных понятиях данный способ применялся лишь в I+0 определениях.

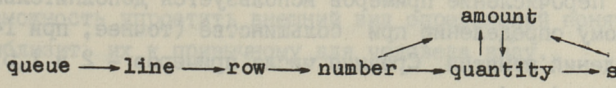
5. Определение, в котором отсутствует есть-понятие

Есть-понятие отсутствует в определении, когда оно не существует (для наиболее универсальных понятий), или когда по каким-то причинам предпочитают определение посредством перечисления свойств понятия. Рассмотрим оба случая.

Определение с помощью примеров. Этот способ применяется для абстрактных понятий, трудно характеризуемых через есть-понятие, например:

(I0) "number - 3, 13, 33 and 103 are numbers".

Понятия, рассматриваемые в данной статье, как правило, не входят в эту категорию. Однако все они приводятся в цепи есть-понятий к некоторым фундаментальным понятиям. Последние же либо определены с помощью примеров, либо их определения образуют замкнутый круг (цикл), т.е. понять их можно опять-таки лишь с помощью примеров. Приведем пример замкнутого круга из [I0]:



Здесь стрелки указывают на есть-понятия.

Отсутствие есть-понятия в предложении. Точнее, есть--понятие заменяется указательным местоимением (в 0+4 определениях). Например, слово "тот" в (II) "Гармонист - тот, кто играет на гармонике" заменяет есть-понятие "человек". Если же рассматривать словосочетание "тот, кто" как указатель на конкретного человека, то имеем в этом случае не есть-понятие, а свободную роль понятия "гармонист".

Этот способ определения применяется, когда есть-понятия не вполне сформированы (например, у детей очень часто встречается такое определение: "карандаш - это то, чем пишут"), или для некоторых распространенных есть-понятий (человек, вещь).

Есть-иерархия или есть-граф? Мы увидели, что у понятия может быть нуль или несколько есть-понятий, и что последовательность есть-понятий может образовать цикл. Это означает, что для выражения отношений между есть-понятиями недостаточно рассматривать есть-иерархию, а необходимо вводить есть-граф. Это - орграф, в котором, как и раньше, дуги направлены от понятия к есть-понятию. Есть граф обладает следующими свойствами:

- число выходящих из вершины дуг ограничено числом есть--понятий одного понятия (обычно не более трех);
- число входящих в вершину дуг ограничено числом понятий в предметной области;
- длина ациклического пути обычно в пределах пяти, максимально десяти дуг.

6. Перечисление примеров в определениях

Человек легче улавливает суть дела, когда она пояснена примерами. Это учитывается и в словарях: примеры служат для иллюстрации к определению понятия, для задания содержания роли и т.п. Рассмотрим подробнее, какие функции выполняли примеры в определениях.

I. Перечисление примеров используется дополнительно к основному определению при большинстве (точнее, при I4+9) определений понятий. Средние числа примеров - 2 и I,5, максимальные 4 и 4.

2. Примеры определяют заполнение роли "по умолчанию" в 2+I случаях. Например, понятие "маломерка" в определении

(I2) "Маломерка - предмет меньших, чем обычно, размеров ... (обычно об обуви, одежде)"

имеет роль "пример". Она заполнена по умолчанию ("обычно") значениями "обувь", "одежда".

3. Примеры используются для задания содержания роли в 2+6 случаях, как в "колпак - головной убор остроконечной, овальной и т.п. формы".

4. Примеры, задающие фундаментальные понятия, были рассмотрены в предыдущем разделе.

7. "Человеческие" и "машинные" определения понятий

Из сказанного выше видно, что определения, задаваемые вычислительной машиной и человеком, весьма часто различны по структуре. Последние в некотором отношении проще (редко используются отрицания, решающие правила в виде уравнений, более пяти ролей и т.п.), а в некотором отношении - сложнее (используется информация, заложенная в есть-понятии и других связанных с определением понятиях; используется богатый комплекс знаний человеком окружающего мира).

Как же человек ухитряется давать определения более простые, чем машина? Очень часто это достигается за счет переноса сложности в межпонятийные связи. Вместо отрицания ("Предельщик - тот, кто ... не борется за ...") используется соответствующее понятие ("Предельщик - тот, кто ... отказывается от борьбы ..."), зачастую образованное с помощью отрицательной частицы "не-" ("Растяпа - несообразительный ... человек"). Решающие правила задаются в виде соответствующих нечетких понятий ("колпачок - маленький колпак"). Число признаков удерживается в нужных пределах также введением, при необходимости, нового есть-понятия.

Таким образом, использование межпонятийных связей дает возможность упростить внешний вид определений понятий и приблизить их к привычному для человека виду.

Интересно отметить, что большинство числовых характеристик образования понятий оказались, по крайней мере в тенденции, весьма близкими в русском и английском языках. Это позволяет предположить универсальность механизма выражения понятий в разных языках.

Заключение

Наиболее распространенным способом определения нового понятия у человека является введение новой роли в есть-понятие; весьма часто используется также заполнение явно указанной в есть-понятии роли. Замена заполненной роли есть-понятия другой, эвристические правила формирования понятий и определение, в котором отсутствует есть-понятие, употребляются сравнительно редко. В большинстве определениях понятий приводятся примеры, которые чаще всего служат для иллюстрации к определению понятия, а реже для задания содержания роли, для задания содержания роли по умолчанию, для задания фундаментальных понятий.

Совокупность связей между понятиями и есть-понятиями образует не иерархию, как обычно принято, а граф. Анализ рассмотренных понятий позволяет предположить универсальность механизма выражения понятий в разных языках.

Литература

I. Бойко В.В., Савенков В.М. Проектирование информационной базы автоматизированной системы на основе СУБД. - М.; Финансы и статистика, 1982.

2. First Scandinavian research seminar on information modelling and data base management // Acta universitatis Tamperensis. - Ser. B, vol. 17. Tampere, 1982. - 445 p.

3. Поспелов Д.А. Логико-лингвистические модели в системах управления. - М.; Энергоиздат, 1981. - 231 с.

4. Building expert systems. / Edited by Frederick Hayes-Roth, Donald A. Waterman, Douglas B. Lenat. - Addison-Wesley Publishing company, Inc, 1983. - 431 p.

5. D.B. Lenat. Theory formation by heuristic searsh // Artificial Intelligence. - 1983. - Vol. 21. - P. 31-59.

6. Ершов А.П. Опыт интегрального подхода к актуальной проблематике программного обеспечения // Кибернетика, -1984,-№ 3. - С. II-2I.

7. Горелик А.Л., Скрипкин В.А. Методы распознавания. - М.; Высшая школа, 1984. - 207 с.

8. Progress in pattern recognition, Volume 1. / Edited by L.N. Kanal and A. Rosenfeld. - Amsterdam: North-Holland Publishing Company, 1981. - 391 p.

9. Словарь русского языка / Под ред. А.П. Евгеньевой. В 4 томах. Государственное издательство иностранных и национальных словарей. - М., I957-I95I. Т. I - 963 с., т. 2 - IOI3 с., т. 3 - 99I с. т. 4 - I080 с.

10. Oxford Advanced Learner's Dictionary of Current English. - Oxford University Press, 1980. - 1036 p.

11. Roussopoulos N. CSDL: a conceptual schema definition language for the design of data base applications // IEEE Transactions on Software Engineering. - September 1979. - Vol. SE-5, N 5. - P. 481-496.

I2. Войшвилло Е.К. Понятие. - М.: Издательство Московского университета, I967. - 285 с.

J. Tepandi

Concept Description for an Information System End-User

Abstract

An end-user of an information system needs clear and natural descriptions of his problem area concepts. To generate such descriptions in the process of inductive learning, one must know the structure of a natural language concept description.

On the basis of English and Russian explanatory dictionaries, the methods of concept descriptions in these languages are observed. In both languages the new concepts are most frequently formed by adding a new slot with a new property to the is-a-concept. Less frequently, an empty slot of the is-a-concept is filled. Replacing a property, heuristics (e.g. "make parts coincide") and definition without an is-a-concept are used rarely. In most definitions, the examples play an important role, illustrating the new concept or (more rarely) defining the new property or a fundamental concept. The importance of introducing is-a-graphs instead of the usual is-a-hierarchies is argued. The frequences of using different concept formation methods in both languages were found to be rather similar.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.03

Я.Я. Тепанди

ЗАГРУЗКА И ПОПОЛНЕНИЕ БАЗЫ ЗНАНИЙ В ДИАГНОСТИЧЕСКОЙ ЭКСПЕРТНОЙ СИСТЕМЕ

Экспертная система предоставляет решения важных проблем на экспертном уровне и является:

- эвристической, т.е. рассуждающей на уровне здравого смысла;
- "прозрачной", т.е. предоставляющей толкование логики рассуждений и ответы на вопросы о ее знаниях;
- развивающейся, т.е. постоянно интегрирующей новые знания вдобавок к существующим [1].

Пополнение базы знаний (БЗ) является узким местом, "горлышком бутылки" для экспертных систем [2]. Оно занимает много времени и может привести к ошибочным БЗ. Многие системы создаются следующим образом: БЗ создается с помощью специалиста по системе; пользователь вводит только данные своей задачи (база данных задачи или "доска"). Система использует их, но не запоминает. Естественно, лучше, если специалист предметной области сам обновляет БЗ своей системы.

Интересный способ пополнения знаний предоставляет игра в загадки на ЭВМ под названием "Животные". В этой игре пользователь задумывает имя животного. Система же задает да-нет вопросы о животном, пока не выдаст свое предположение. Если это предположение не удовлетворяет пользователя, то система запрашивает правильное название, дополнительный да/нет вопрос, различающий предположение системы и название пользователя, и ответ на этот вопрос. Эти данные запоминаются и в следующий раз ими можно будет воспользоваться. Да/нет вопросы для коммуникации с пользователем применяются, например, в экспертной системе TAXADVISOR [3].

В описываемой ниже системе ТООТС объединены основные идеи игры "Животные" и системы TAXADVISOR. База знаний си-

стемы организована в виде расширенного бинарного дерева (РБД) [4]. В первой части статьи рассматриваются операции на РБД, во второй - структура и свойства системы ТООТС, в третьей - принципы использования РБД в этой системе.

I. Операции на расширенном бинарном дереве

Определим РБД как бинарное дерево, полученное путем присоединения к некоторому исходному бинарному дереву внешних узлов (листьев) [4]. Будем считать дерево, состоящее из одного листа, расширением пустого дерева (рис. I, а).

Пометим каждый узел у РДБ элементом э(у) из некоторого множества S . Заметим, что каждый узел РБД имеет точно два потомка. Легко показать, что число листьев РБД на единицу больше числа его внутренних вершин. Определим следующие операции:

- ИНИТ (к): инициализируется РБД, т.е. создается РБД, состоящее из единственного листа к;

- ВСТАВИТЬ (а,в,с,н): вместо листа а вставляется внутренний узел в, притом его потомками будут лист с - в направлении $H \in \{$правый, левый$\}$ и лист а - в противоположном направлении;

- ПОСТРОИТЬ (а,л,п): строится левый и правый сыновья листа а.

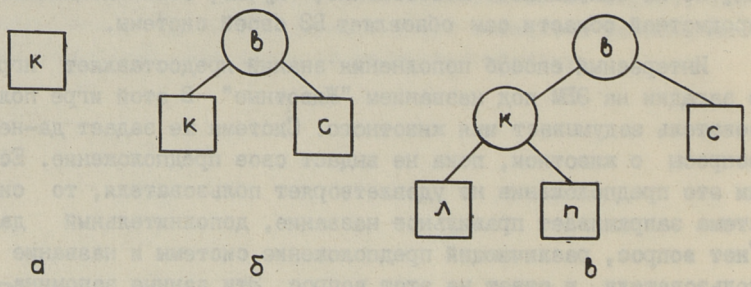


Рис. 1.

На рис. I, а, б, в показаны результаты последовательного применения операций ИНИТ (к), ВСТАВИТЬ (к, в, с, правый) и ПОСТРОИТЬ (к,л,п).

Для дальнейшего представляет интерес выяснить, как строить заданное РБД T, используя описание операции. С помощью операций ИНИТ и ПОСТРОИТЬ это делается просто: корень инициализируется, а внутренние вершины дерева T проходят некоторым систематическим способом сверху вниз, при этом в каждой вершине строятся ее сыновья.

Для построения дерева T с помощью операций ИНИТ и ВСТАВИТЬ можно воспользоваться следующим алгоритмом.

Алгоритм А. Построение РБД с помощью операций ИНИТ и ВСТАВИТЬ.

Вход. РБД T с n внутренними вершинами.

Выход. Последовательность П, состоящая из операций ИНИТ и ВСТАВИТЬ.

Метод

I. Поставить каждой внутренней вершине b дерева T в соответствие лист $n(b)$, полученный движением к левому сыну вершины b и (если это не лист) вниз-направо, насколько возможно (рис. 2, а).

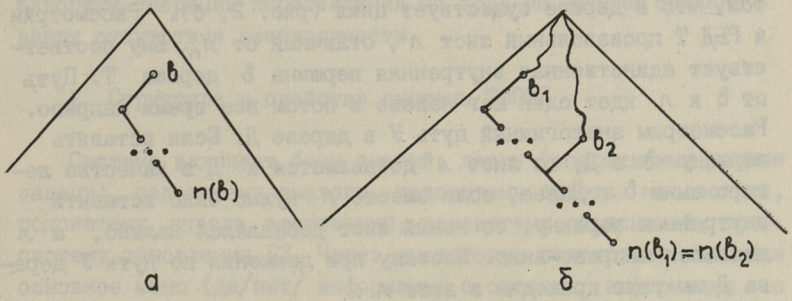


Рис. 2.

2. Пронумеровать внутренние вершины T в прямом порядке (рекурсивно корень, левое поддерево, правое поддерево) (рис. 3, г). Обозначим вершины в этом порядке через $b_1,\ldots,b_n$. Обозначим самый правый лист кроны T через λ_n.

3. Выдать первый элемент последовательности П – операцию ИНИТ (λ_n). Сконструировать новое дерево Д, применяя указанную операцию. Сопоставить λ_n с корнем дерева T.

4. Повторить для $i = 1, \ldots, n$: двигаться в дереве Д до листа b'_i, соответствующего вершине b_i в дереве Т; выдать операцию ВСТАВИТЬ $(b'_i, b_i, n(b_i)$, левый) в качестве следующего элемента последовательности П; применить эту операцию к Д и принять результат за новое значение Д. Вершина, которая вставилась вместо корня Д, будет сопоставляться с корнем Т.

Конец алгоритма

Покажем, что результатом выполнения операций последовательности П при данном порядке является дерево Д. Для этого заметим, что исходные для Т и Д бинарные деревья совпадают по конструкции. Далее лист Λ_n, введенный самым первым в дерево Д, передвигался все время направо в результате операций ВСТАВИТЬ (..., левый), оказавшись самым последним и в кроне дерева Д.

Заметим, что на шаге I алгоритма с каждой внутренней вершиной дерева Т сопоставляется единственный лист, и напротив: каждому листу ставится в соответствие единственная вершина. Действительно, предположение о наличии двух различных вершин b_1 и b_2, для которых $n(b_1) = n(b_2)$, приводит к неправильному выводу о том, что левые сыновья вершин b_1 и b_2 должны находиться на одном пути, а следовательно, о том, что в дереве существует цикл (рис. 2, б). Рассмотрим в РБД Т произвольный лист Λ, отличный от Λ_n. Ему соответствует единственная внутренняя вершина b дерева Т. Путь от b к Λ идет один шаг налево и потом все время направо. Рассмотрим аналогичный путь У в дереве Д. Если вставить вершину b в Д, то лист Λ добавляется к Д в качестве левого сына b. Далее, если вместо Λ нужно было вставить внутреннюю вершину, то новый лист добавлялся налево, а Λ двигался направо-вниз. Поэтому при движении по пути У дерева Д мы тоже приходим в лист Λ.

Пример. Дерево на рис. 3, г порождается следующей последовательностью:

ИНИТ (Λ_7)
ВСТАВИТЬ ($\Lambda_7, b_1, \Lambda_1$, левый)
ВСТАВИТЬ ($\Lambda_1, b_2, \Lambda_2$, левый)
ВСТАВИТЬ ($\Lambda_7, b_3, \Lambda_3$, левый)
ВСТАВИТЬ ($\Lambda_3, b_4, \Lambda_4$, левый)

ВСТАВИТЬ ($л_3$, $в_5$, $л_5$, левый)
ВСТАВИТЬ ($л_7$, $в_6$, $л_6$, левый)

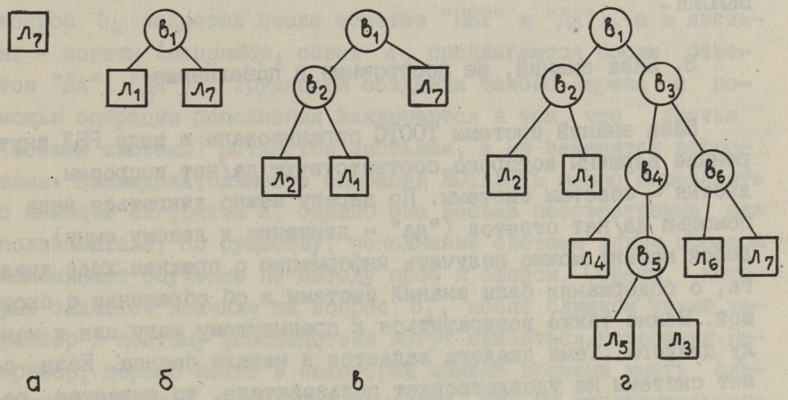


Рис. 3.

На рис. 3, а, б, в приведены некоторые шаги построения Д.

На первый взгляд может показаться, что операция "двигаться до листа $в'_i$ ..." на шаге 4 алгоритма очень неудобная; однако, это как раз то, что делает пользователь, когда выполняет операцию ВСТАВИТЬ. Можно было бы явно включить в алгоритм операции передвижения по РБД, но это не сделано ввиду отсутствия необходимости.

2. Структура и свойства системы ТООТС

Система включает базу знаний, доску (оперативные данные задачи), подсистему диалога, подсистему ответа на запрос, подсистему вывода заключений, подсистему объяснений, подсистему обновления БЗ. Через диалоговую подсистему задается основное меню (да/нет/ информация о системе/ информация о ходе данного разговора/движение назад/ движение на начало/ конец сеанса) и вводится ответ пользователя. На простые ответы пользователя (информация о системе, объяснения, переход на начало, конец сеанса) подсистема реагирует самостоятельно, но обычно происходит обращение к подсистеме ответа на запрос. Последняя использует подсистему вывода заключений для движения по БЗ, согласно ответам пользователя.Пользователь может получить общую информацию о системе, о ходе диалога и о содержании БЗ.

Система реализована на языке БЭЙСИК под ОСRSX-IIM ЭВМ СМ-4 и используется для консультации начинающих по языку БЕЙСИК.

3. База знаний, ее построение и пополнение

База знаний системы ТООТС организована в виде РБД, внутренние вершины которого соответствуют да/нет вопросам, а листья - советам системы. По дереву можно двигаться вниз с помощью да/нет ответов ("да" - движение к левому сыну). В любой момент можно получить информацию о прежнем ходе диалога, о содержании базы знаний системы и об обращении с системой. Можно также возвратиться к предыдущему шагу или к началу диалога. Тема диалога задается в начале сеанса. Если совет системы не удовлетворяет пользователя, то выдается рекомендация узнать правильный совет и занести его в систему, чтобы в следующий раз можно было им воспользоваться.

Пусть $б_1$ - вопрос, $л_1$ и $л_7$ - советы, $б_2$ - новый вопрос, $л_2$ - новый совет. Тогда пример добавления нового совета в БЗ приведен на рис. 3 б, в (здесь ответ "да" на новый вопрос приводит к выдаче нового совета).

В ходе пополнения БЗ системы выяснились следующие обстоятельства:

- пополнение БЗ самим пользователем имеет то преимущество, что наиболее часто возникающие проблемы будут требовать более краткого диалога для выдачи рекомендации;

- пополнение БЗ самим пользователем имеет тот недостаток, что знания в БЗ могут оказаться противоречивыми, недостаточно понятными или будут неэкономно организованы;

- для создания БЗ с заранее установленной структурой операция ВСТАВИТЬ неудобна.

Остановимся подробнее на последнем выводе. На самом деле операция ВСТАВИТЬ - весьма частный вид пополнения РБД, поэтому удивление вызывает не то, что она оказывается в некоторых случаях недостаточной, а то, что во многих случаях она оказывается весьма удобной. Пользователи легко воспринимают основную идею и охотно ее применяют. Сложности же возникают тогда, когда этим методом пытаются создавать заранее спроектированную БЗ. Покажем это на примере (рис. 3,г)

Во внутренних вершинах представлены вопросы (например, вопрос $в_4$ задается после ответов "НЕТ" и "ДА"), а в листьях - советы (например, совет $л_2$ предлагается после ответов "ДА", "ДА"). Трудности создания такого дерева с помощью операции пополнения заключаются в том, что листья (советы системы) останутся листьями, а не заменятся вопросами. Последовательность операций ВСТАВИТЬ можно определить с помощью алгоритма А, однако она весьма неестественна для пользователя. По существу, пополнение системы таким образом напоминает обучение по методу проб и ошибок. Советы, которые задаются вначале на вопрос $в_1$, носят слишком узкий характер и поэтому впоследствии могут оказаться неверными. Например, дерево может в некоторый момент времени иметь следующий вид (рис. 3,б). Это означает, что из утвердительного ответа на $в_1$ следует совет $л_1$. После добавления вопроса $в_2$, утвердительный ответ на вопрос $в_1$ может уже привести к совету $л_2$ (рис. 3,в). Имеем пример немонотонного рассуждения.

Вследствие этого, в системе ТООТС используются три режима изменения БЗ: загрузка, инициализация и пополнение. Первый режим предназначен для ввода БЗ программистом непосредственно в виде РБД. Второй режим используется пользователем, который желает сам ввести свою БЗ, поэтому при инициализации РБД задается корень и два его сына. Режим пополнения реализован с помощью операции ВСТАВИТЬ и предназначен для окончательной настройки БД пользователем.

Различия между пополнением и загрузкой БЗ иллюстрируют различия между проектированием БД снизу-вверх и сверху-вниз, между пополнением знаний в ходе практической работы и систематического обучения, между немонотонными и монотонными рассуждениями. Они аналогичны соответствующим различиям в технологии проектирования систем программирования.

Литература

1. Buchanan B.G., Duda R.O. Principles of rule-based expert systems // Advances in Computers. Vol. 22. New York: Academic Press, Inc., 1983. - P. 163-216.

2. Raulefs P. Expert systems: state of the art and future prospects // GWAI 81. - Berlin: Springer-Verlag, 1981. - P. 98-111.

3. Michaelson R., Michie D. Expert systems in business // Datamation. - November 1983. - Vol. 29, N 11. - P. 240-246.

4. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. - М.: Мир, 1980. - 476 с.

J. Tepandi

Loading and Updating the Knowledge Base of a Diagnostical Expert System

Abstract

An expert system shell prototype TOOTS for applications with hierarchical knowledge structure is discussed. Using the TOOTS system, one may seek for advice, insert new advices in the dialogue mode, load the knowledge base (KB) in the dialogue mode, load the KB in the edit mode. An algorithm for loading a previously designed KB in the dialogue mode is given.

№ 645
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
УДК 681.3.06

З.А. Йокк, Э.Х-Т. Нунапуу

РАЗРАБОТКА ИНТЕРФЕЙСА МЕЖДУ ЯЗЫКОВЫМ ПРОЦЕССОРОМ И СУБД В ГЕНЕРАТОРНОЙ СИСТЕМЕ ОБРАБОТКИ ДАННЫХ ГЕНСИ

В современных системах обработки данных обязательным компонентом является система управления базами данных (СУБД). Использование СУБД в качестве ядра системы обработки данных требует решения проблемы разработки интерфейса между базой данных и языками обработки данных. В статье рассматривается проблема интерфейса в более общей постановке, чем в обычных системах СУБД с базовым языком [1].

Авторы данной статьи ставили перед собой задачу разработки универсального интерфейса, обеспечивающего использование разнотипных баз данных. Суть предлагаемого решения заключается в том, что единицей обмена информации между базой данных и языком обработки данных служит так называемая суперзапись, которой в базе данных может соответствовать несколько взаимосвязанных записей.

В начале статьи проблема разработки интерфейса рассматривается в общей постановке. Затем приводится реализация интерфейса в системе обработки данных ГЕНСИ.

1. Организация интерфейса между языковым процессором и СУБД

Разработка интерфейса между языковым процессором генераторной системы ГЕНСИ и СУБД является одной из центральных задач реализации системы в силу следующих основных причин.

1. В настоящее время одной из наиболее продуктивных идей является интеграция неоднородных баз данных с целью обеспечения одновременного и совместного использования в рамке одной информационной системы нескольких СУБД. Предлагаемый

в статье универсальный интерфейс позволяет преодолеть программную и информационную несовместимость разнотипных баз данных.

2. Процедуры (функции) обработки данных, представляемые внешнему пользователю системой ГЕНСИ для решения прикладных задач, являются высокоуровневыми, независимыми от среды хранения структур физических данных. Достигается это путем разделения логического и физического уровней. Разработка интерфейса между названными уровнями и является главной целью нестоящей статьи. Данная постановка задачи аналогична постановке задачи в работе [2]. Выбранный нами подход решения задачи существенно упрощен и приближен к реальным ситуациям. Пользователь сам задает (программирует) преобразователь физического уровня базы данных к логическому уровню.

Основная идея разрабатываемого интерфейса заключается в использовании в качестве единицы обмена так называемой суперзаписи. Суперзапись - это линейный список показателей, который является либо ответом на запрос извлечения данных из базы, либо совокупностью данных, передаваемых прикладной программой в базу. В общем случае одной суперзаписи соответствует совокупность записей нескольких типов записей базы данных. Это значит, что для получения суперзаписи необходима предварительная обработка данных.

На рис. I приведена упрощенная схема реализации интерфейса между базой данных и языком пользователя (прикладной программой).

Для доступа к элементам базы данных прикладной программист должен составить описание суперзаписи. Это описание используется при передаче данных из базы в специальную область данных программы. Передачу данных в эту область программы осуществляет системная программа, названная на рис. I преобразователем.

В результате реализации интерфейса по приведенной схеме (см. рис. I) достигается высокая степень независимости прикладных программ от данных. При необходимости можно менять не только схему базы данных, а даже систему управления базами данных. В последнем случае необходимо воспользоваться другим преобразователем.

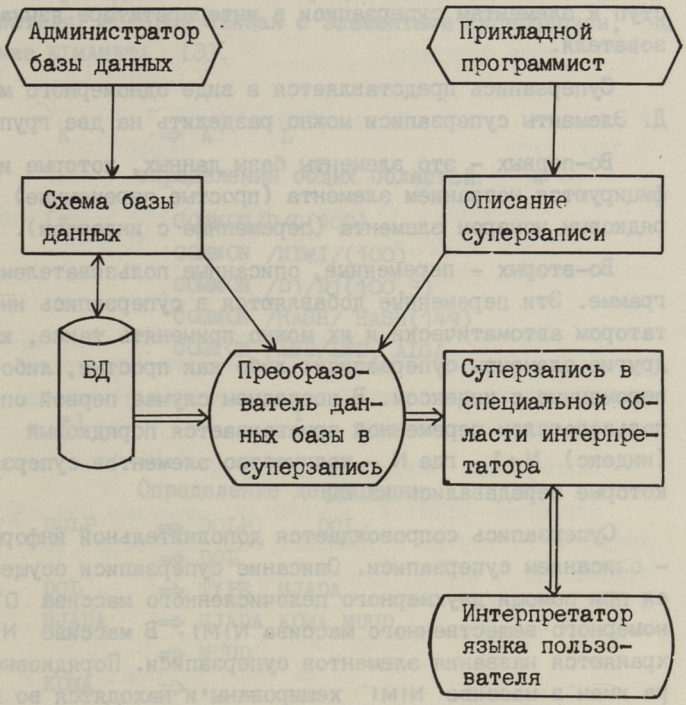


Рис. 1. Схема реализации интерфейса между базой данных и интерпретатором.

2. Применение суперзаписи в генераторной системе обработки данных ГЕНСИ

Необходимым компонентом системы ГЕНСИ является язык пользователя, в котором существует возможность применения логических условий и арифметических выражений. В данном случае этот язык реализован при помощи системы построения трансляторов ЭЛМА в виде интерпретатора.

Для вычисления логических условий или арифметических выражений необходимы данные, хранящиеся в базе данных (БД) пользователя. Управляющая программа системы ГЕНСИ считывает данные из базы, преобразует их в суперзапись и передает в общую область интерпретатора.

Далее рассматривается, как практически реализован доступ к элементам суперзаписи в интерпретаторе языка пользователя.

Суперзапись представляется в виде одномерного массива Д. Элементы суперзаписи можно разделить на две группы.

Во-первых - это элементы базы данных, которые идентифицируются названием элемента (простые переменные) или порядковым номером элемента (переменные с индексом).

Во-вторых - переменные, описанные пользователем в программе. Эти переменные добавляются в суперзапись интерпретатором автоматически и их можно применять также, как и другие элементы суперзаписи: либо как простые, либо как переменные с индексом. В последнем случае первой описанной пользователем переменной присваивается порядковый номер (индекс) N+1, где N - количество элементов суперзаписи, которые передавались из БД

Суперзапись сопровождается дополнительной информацией - описанием суперзаписи. Описание суперзаписи осуществляется при помощи двухмерного целочисленного массива D1 и одномерного вещественного массива NIMI. В массиве NIMI сохраняются названия элементов суперзаписи. Порядковые номера имен в массиве NIMI хешированы и находятся во вспомогательном массиве HASH.

В массиве D1 хранится дополнительная информация об элементах суперзаписи - тип, длина и относительный адрес, где

D1(i,1) - тип i-го элемента,
D1(i,2) - длина i-го элемента,
D1(i,3) - адрес i-го элемента.

Тип элемента - это целое число от I до 5, значение которого приведено в таблице I.

Таблица I

Типы элементов суперзаписи

Число	Тип элемента	Длина элемента
I	целый	2
2	вещественный	4
3	литерный	длина текста в символах
4	вещественный	8
5	целый	4

Далее приведена соответствующая часть описания языка пользователя, который связан с элементами суперзаписи, на метаязыке ELMAMETA [3].

```
W  10      K        ⇒ A      L
 #                   определение общих областей      #
          [*           COMMON/D/D(100)
                       COMMON /NIMI/(100)
                       COMMON /D1/D1(100,3)
                       COMMON /HASH/ HASH(149)
                       COMMON /IND/IND, AIGA
          *]                 ⋮

#                     Определение декларации          #
W   50    DCLP       ⇒ DCLP     DCL
W   55               ⇒ DCL
W   60    DCL        ⇒ TYPE  MJADA
W   65    MJADA      ⇒ MJADA KOMA MUUD
W   70               ⇒ MUUD
W   75    KOMA       ⇒ ,
W   80    MUUD       ⇒ IDENT

#                     Хеширование номера декларированного
                      элемента                        #
          [*
          CALL PIIRID (IDENT, VAL, ID)
          CALL KOHASH (ID, ADR, NIMI, HASH, AIGA)
          J = HASH (ADR)
          IF (J ⌝ = 0) GOTO 1058
          IND = IND + 1
          HASH (ADR) = IND
          NIMI (IND) = ID
          D1 (IND, 1) = T
          D1 (IND, 2) = P
```

```
            D1 (LND, 3) = SADR
            SADR = SADR + P
       1058 CONTINUE
                  *]
#                  Определение типов                  #
W 100  TYPE                              TYPE1
     [*  LOE = LOE + 1
         IF (LOE = 1) SADR = D1 (LND, 3) + D1 (LND, 2)
         T = TYPE1.T
         P = TYPE1.P
       *]
```

Поскольку СПТ ЭЛМА основывается на атрибутной технике, то информация передается в суперзапись при помощи семантических атрибутов в виде

переменная. название атрибута

Атрибуты TYPE1.T и TYPE1.P передают целочисленные значения атрибутов и означают соответственно тип и длину переменной #

```
W 110                          ⇒ TYPE2
        [* ...                  *]
W 115                          ⇒ CHAR * KONST
        [* ...                  *]
W 120   TYPE1                  ⇒ INTEGER
        [* TYPE1.P = 4
           TYPE1.T = 5
         *]
```

Длина переменной 4, тип переменной 5

```
W 260   ARITEX                 ⇒ ARITEX OPER1 TERM
```

Вычисление значения арифметического выражения

```
  [*    CALL ARITEX (ARITEX.V, TERM.V,
        ARITEX.V1, TERM.V1, OPER1.FIXT,
        ARITEX.FIXT, TERM.FIXT)      *]

W 265                          ⇒    TERM
W 270                          ⇒ -  TERM
  [*    IJ1 = TERM.FIXT
        IF(IJ1 = 1!IJ = 5) GOTO 1272
```

```
 1272    ARITEX.V = - TERM.V
         ARITEX.FIXT = IJ1
         GOTO 10272
 2272    IF (IJ1 = 2) GOTO 3272
         GOTO 10272
 3272    ARITEX.V1 = - TERM.V1
         ARITEX.FIXT = IJ1
10272    CONTINUE
   *]
W  300   ASS      ⇒ MUU  =  ARITEX

#  Присваивание с преобразованием типа        #
  [*
                :
         J = MUU.V
         I = D1 (J,1)
         K = D1 (J,3)
         P1 = D1 (J,2)
         L = ARITEX.FIXT
         GOTO (3300,4300,5300,10300,7300), I
         GOTO 10300
 3300    IF (L=2) ARITEX.V = ARITEX.V1
         112 = ARITEX.V
         CALL 12 (D,112)
   *]    GOTO 10300
```

Здесь применяются подпрограммы переноса данных в суперзапись. (I2 - подпрограмма переноса целых чисел длиной 2 байта)

Разработанный универсальный интерфейс реализован в рамках системы ГЕНСИ, которая эксплуатируется несколькими предприятиями Эстонии.

Литература

I. Мартин Дж. Организация баз данных в вычислительных системах.-М.: Мир, 1980. - 486 с.

2. Калиниченко Л.А. Методы и средства интеграции неоднородных баз данных. М.: Наука, 1983. - 423 с.

3. Вооглайд А.О., Лепп М.В., Лийб Д.Б. Входные языки системы ЭЛМА // Тр. Таллинск. политехн. ин-та. - 1982. - № 524. - С. 79-96.

V. Jokk, E. Õunapuu

An Interface between Compiler Writing System ELMA and Data Processing System GENSI

Abstract

In this paper a problem of creating the interface between programs created by compiler writing system ELMA and data processing system GENSI is considered. First general ideas are discussed, after that problems of the interface creating are described.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.06

Л.К. Выханду, В.А. Йокк,
Э.Х.-Т. Нунапуу

О ТЕХНОЛОГИЗАЦИИ ПОСТРОЕНИЯ ПРИКЛАДНЫХ СИСТЕМ ОБРАБОТКИ ДАННЫХ

Рабочая группа кафедры обработки информации Таллинского политехнического института имеет многолетний опыт создания систем обработки данных (СОД), используемых в разных областях (экономики, медицины, социологии и т.д.). За это время выработаны следующие принципы создания СОД:

1) максимально использовать существующие пакеты программ (системы управления базами данных (СУБД), пакеты статистики, экранный редактор) и минимально программировать самому (принцип ленивого программиста);

2) если уж программировать самому, то предельно универсальные технологические средства (генераторы ввода-вывода, генераторы отчетов и т.д.);

3) использовать существующие инструментальные системы (системы построения трансляторов (СПТ));

В настоящей статье рассматриваются проблемы применения названных инструментально-технологических средств при разработке систем обработки данных (информационных систем). Во-первых, задается общая схема применения технологических средств в разработанной нами технологии построения прикладных систем обработки данных (рис. 1). Во-вторых рассматриваются проблемы, связанные с синтезом СПТ и СУБД в системе обработки данных.

Использование СПТ и СУБД в СОД требует решения следующих проблем:

1) применения СПТ в СОД;
2) организация интерфейса между языковым процессором и разными СУБД.

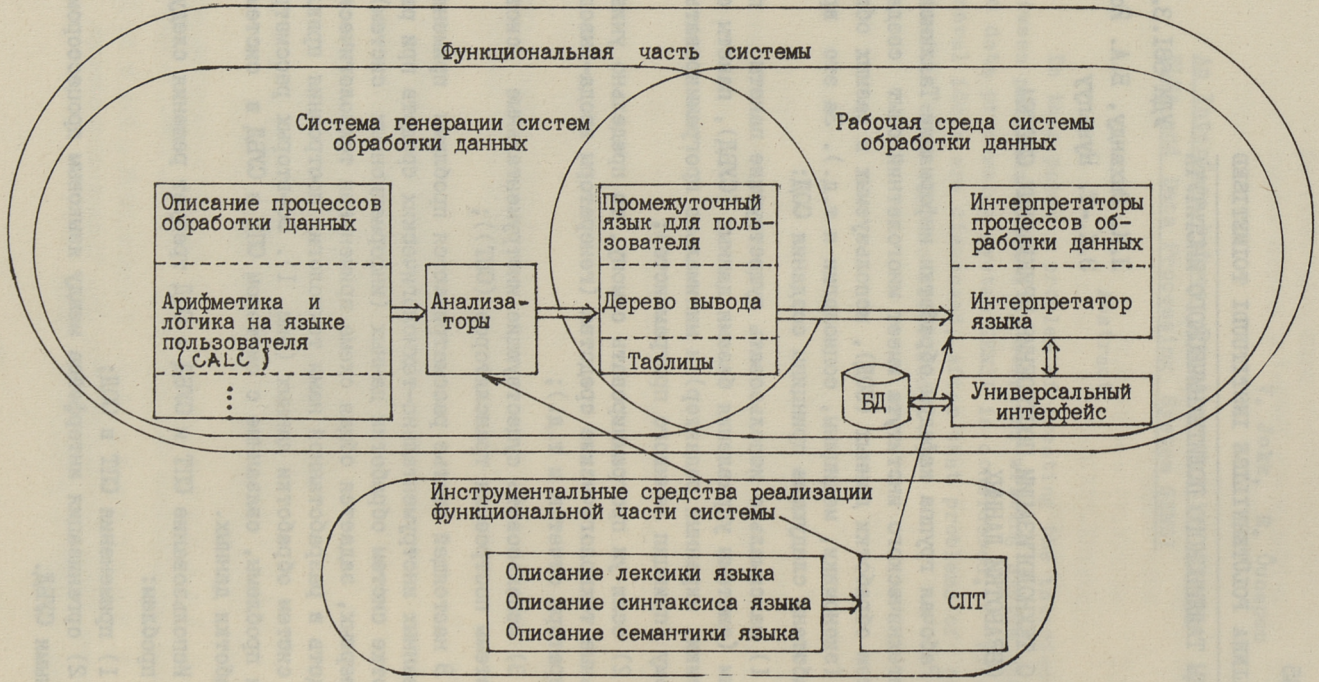


Рис. 1. Общая схема применения технологических средств.

В результате повысится как производительность программиста, так и гибкость и надежность СОД.

I. Использование СУБД и СПТ в СОД

Системы управления базами данных можно при создании СОД использовать как подсистему СПТ. Проблемы, связанные с таким применением СУБД в СПТ рассматривались в [I]. Поскольку подобное использование СПТ и СУБД, в нашем случае не оправдано, рассмотрим далее, какие результаты дает применение СУБД и СПТ в качестве инструментальных средств реализации СОД. В данном случае СУБД используется для выполнения функции хранения и манипулирования данными как обычно, а возможности применения СПТ рассматриваются ниже.

Применение СПТ в СОД в принципе возможно в двух вариантах:

I) СПТ как средство разработки конкретных языков СОД и построения транслятора для этих языков;

2) СПТ как инструментальная система реализации технологических средств СОД.

Первое направление применения СПТ более привычно. По нему в нашем институте реализованы такие системы, как АСТА [2] и ПАРЕС [3].

Для наших целей более подходящим представляется второе направление. По сравнению с общепринятым подходом в таком варианте с применением СПТ возникает целый ряд особенностей:

I) в одной СОД можно использовать несколько разных языков;

2) созданный язык используется в разных подсистемах СОД (например, язык логических условий обычно используется в 7-8 местах);

3) данные управляют не только работой программ, составленных на реализованном языке, но работой всей создаваемой СОД;

4) работа в диалоге накладывает новые требования на создаваемый язык: одну часть информации получают в диалоге, другую из языка;

5) использование создаваемых языков делает необходимым координацию совместной работы программ, написанных на этих

языках и используемых готовых инструментальных средств и функциональных пакетов.

2. Технология применения СПТ в СОД

Исходя из концепций, на которых основывается технология создания СОД, можно сказать, что СПТ целесообразно использовать:

1) для создания технологических инструментальных средств СОД;

2) для реализации языков управления.

Одним из важнейших технологических инструментальных средств СОД является интерпретатор языка логических условий и арифметических выражений. Этот интерпретатор языка применяется практически во всех подсистемах СОД. Так, при вводе данных при помощи названного языка задаются условия, которым должны соответствовать данные. Выбор данных из базы также осуществляется языком логических условий. При генерации отчетов условия строк и столбцов отчетов, а также нужные при составлении отчета вычисления задаются с помощью вышеназванного языка.

Для создания языков управления СПТ применяется как средство разработки конкретного языка.

3. Интерфейс между СУБД и интерпретатором языка

Применение СПТ как инструментальной системы реализации технологических средств СОД подразумевает решение проблемы организации интерфейса между интерпретатором и СУБД, т.е. спецификацией интерфейса со средствами СПТ. Выработанный нами интерфейс является универсальным и не зависит от используемой СУБД. Тем самым отпадают многие теоретические проблемы, связанные с совместным применением СУБД разных типов [4]. Это достигается тем, что манипулируемые данные преобразуются в стандартный вид, так называемую суперзапись. Суперзапись представляет собой последовательность полей, которая сопровождается локализирующей информацией (тип, длина и относительный адрес) и индикаторами базы данных. Последние управляют работой того или иного блока программы.

Реализацию интерфейса рассматривают подробнее в статье "Разработка интерфейса между языковым процессором и СУБД в системе обработки данных ГЕНСИ" данного сборника (см. с.25).

Реализация

По описанной методике реализован язык, который применяется для реализации входных и выходных генераторов настраивающейся СОД ГЕНСИ.

Язык реализован так, что конструктор языка работает на ЕС-1055, а результат работы переносят при помощи магнитной ленты на СМ-4. Система ГЕНСИ, работающая на СМ-4, внедрена в Таллинском научно-производственном объединении нетканых материалов "Мистра" и в производственном швейном объединении "Балтика".

Заключение

Анализ целесообразности выбранной методики применения СПТ приводит к следующим выводам:

1) производительность труда программиста при создании настраивающейся СОД повышается примерно в 5-6 раз, а результат работы вполне сравним с тем, который получается при программировании вручную;

2) заметно повышается гибкость СОД (легко вносить изменения в создаваемые языки, модифицировать используемую СУБД и т.д.);

3) так как все языки (арифметика, логические выражения, ...) и все используемые пакеты уже отлажены и надежны, то отладка СОД состоит в диалоге на очень высоком уровне и программист работает как навигатор, а не как программист в обычном смысле этого слова.

Литература

I. Тепанди Я.Я. О проблемах применения системы управления базами данных в СПТ // Автоматизация производства пакетов прикладных программ (автоматизация производства трансляторов): Тезисы докладов.- Таллин, 1980. - С. 183-187.

2. Микли Т.И., Крахт В.А., Выхандуу Л.К. Базовая система для создания АИПС ВШ Эстонской ССР / НИИВШ ЭКСПРЕСС ИНФОРМАЦИЯ. - Москва, 1976. - 19 с.

3. Крахт В.А. Эйвак Ю.Э., Вассиль М.Х. Система Парес: Архитектура и возможности // Программирование. - 1984.-№ 5. - С. 75-82.

4. Калиниченко Л.А. Методы и средства интеграции неоднородных баз данных. - М.: Наука, 1983. - 423 с.

L. Vyhandu, V. Jokk, E. Õunapuu

About Technology to Build Application Systems of Data Processing

Abstract

In this paper the problems arising in building technological means for systems of data processing are considered. To create these means the translation writing system is used in an untraditional way.

№ 645
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
УДК 681.3

М.П. Кривенко, И.В. Мацкевич,
Л.К. Выханду

СТАТИСТИЧЕСКИЙ АНАЛИЗ ТРЕНДА

Актуальной в теоретическом и прикладном планах является задача анализа случайных процессов с изменяющимися за период наблюдения характеристиками. Особый интерес при разработке статистической модели в такой ситуации представляет случай, когда в модель вкладывается по возможности меньше априорной информации, например, предполагается только, что значение параметра возрастает за период наблюдения, без уточнения закона изменения. Основной довод в пользу такого подхода заключается в том, что чаще всего нет достаточно полной информации для построения жестких адекватных моделей, и использование любого более узкого класса моделей (например, изменения носят линейный характер) рождает сомнения, а будет ли он работать в иной ситуации, чем наблюдаемая. Сведение к минимуму априорной информации в большей степени характерно разведочному, нежели чем подтверждающему анализу (см. терминологию [I]). Но именно на пути обобщения статистической модели лежит объединение двух указанных типов анализа.

В данной работе предполагается, что изменение характеристик наблюдаемой последовательности есть неизвестная монотонная функция номера наблюдения. Разрабатываемые на основе таких достаточно общих априорных предположений методы нацелены как на предварительный анализ данных (например, фрагментация выборки на участки постоянства, возрастания и убывания значений тех или иных характеристик последовательности), так и на обоснование полученных иным способом результатов (например, анализ остатков при выбранном способе описания динамического процесса).

Постановка задач анализа монотонного тренда выглядит следующим образом. Пусть $x_1, x_2, \ldots, x_N$ – наблюдения независимых случайных величин $X_1, X_2, \ldots, X_N$, каждая из которых имеет распределение, заданное с точностью до параметра λ в виде плотности распределения $f(u, \lambda_i)$ для $i = 1, 2, \ldots, N$. Наличие тренда означает, что набор значений λ принадлежит множеству

$$\Lambda_N = \{(\lambda_1, \lambda_2, \ldots, \lambda_N): \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N\}.$$

Нулевая гипотеза об отсутствии изменений имеет вид

$$H_0: \lambda_1 = \lambda_2 = \ldots = \lambda_N.$$

Конкурирующая гипотеза, предполагающая наличие монотонного тренда, формулируется так

$$H_1: \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N \quad \text{и} \quad \lambda_1 < \lambda_N.$$

Очевидно, что симметричная постановка задачи $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ принципиально ничего не меняет.

В основу решения последующих задач положен принцип максимума правдоподобия. Это означает, что узловой становится задача оценки тренда, приобретающая вид: найти отображение $\vec{\lambda}^*$ выборочного пространства в Λ_N, удовлетворяющее соотношению:

$$\sup_{\vec{\lambda} \in \Lambda_N} L(\vec{\lambda}) = L(\vec{\lambda}^*),$$

$$L(\vec{\lambda}) = \prod_{i=1}^{N} f(x_i, \lambda_i).$$

При этом внимание должно быть сосредоточено на разработке эффективных процедур обработки информации, позволяющих в реальной обстановке получать результаты.

I. Задачи оценки тренда и проверки гипотез

Доказать существование и единственность решения задачи оценки и построить саму оценку удается при соблюдении условия, гарантирующего существование экстремума на любой границе всевозможных значений $\vec{\lambda}$. В основе построения оценки $\vec{\lambda}^*$ лежит следующая идея: если для точки безусловного экстремума не выполняются некоторые ограничения, то искомая точка максимума лежит на границе множества точек, удовлетворяющих этому ограничению. В силу этого решение возникающей задачи нелинейного программирования при наличии ограничений приобретает вид алгоритма. Введя обозначения:

$I = (i_1, i_2, \ldots, i_{K-1})$, где $1 \le i_1 \le i_2 \le \ldots \le i_{K-1} < N$,

$\Gamma_{i,j} = \{(\lambda_i, \lambda_{i+1}, \ldots, \lambda_j): \lambda_i = \lambda_{i+1} = \ldots = \lambda_j\}$,

$\Gamma_I = \Gamma_{1,i_1} \times \Gamma_{i_1+1,i_2} \times \ldots \times \Gamma_{i_{K-1}+1,N}$,

сформулируем в итоге результат, играющий центральную роль в решении задачи оценки.

Теорема. Существует единственное решение $\vec{\lambda}^*$ задачи $\sup_{\vec{\lambda} \in \Lambda_N} L(\vec{\lambda}) = L(\vec{\lambda}^*)$, которое находится с помощью алгоритма:

I) положить $I = (1, 2, \ldots, N-1)$;

2) найти решение $\vec{\lambda}^I$ задачи $\max_{\vec{\lambda} \in \Gamma_I} L(\vec{\lambda}) = L(\vec{\lambda}^I)$;

3) закончить работу алгоритма, если $\vec{\lambda}^I \in \Lambda_N$;

4) построить новое значение I , объединяя в группы те наблюдения, оценки параметров для которых не удовлетворяют требуемым ограничениям; перейти к выполнению шага 2.

Решение задачи оценки в частном случае одномерного параметра сформулировано в [2].

Здесь целесообразно привести пример, иллюстрирующий вид оценки. Пусть числа x_i - наблюдения последовательности нормально распределенных случайных величин, которой свойствен тренд среднего при постоянной дисперсии. Оценка максимального правдоподобия для последовательности наблюдений

88, 275, 39, 64, 283, I77;

примет вид

88, I26, I26, I26, 230, 230.

Графически оценку удобно изобразить в виде ступенчатой функции:

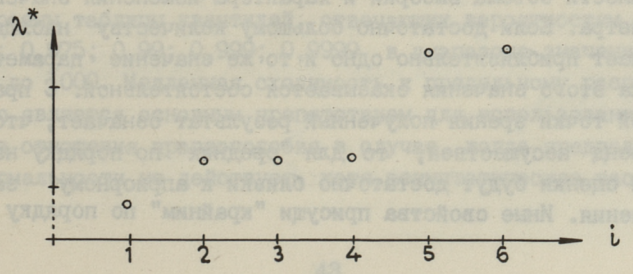


Рисунок служит иллюстрацией для понятий, введение которых создаст определенные удобства. Назовем группы одинаковых элементов оценки ступеньками. Тогда

k – число ступенек,

m_i – длина i-й ступеньки,

h_i – высота i-й ступеньки для $i = 1,2,\ldots,k$.

Подчеркнем, что ступенька соответствует участку постоянства значений параметра. Разбиение на участки постоянства отвечает максимуму правдоподобия и минимуму среднеквадратичного отклонения.

Выборочные свойства оценки представляют как самостоятельный интерес, так и играют существенную роль при построении решающих прецедур. При этом удается описать такие характеристики оценки, как распределения крайних ступенек, числа ступенек.

Доказано [3], что распределение числа ступенек определяется следующим образом:

$$p(N,i) = Pr\{K = i, N\} = |S_N^i|/N!,$$

где S_N^i – числа Стирлинга I-го рода.

Известны асимптотические свойства этого распределения:

при $N \to \infty$ $K \to N(\ell nN, \ell nN)$.

Если распределение числа ступенек в основном необходимо в плане решения задачи проверки гипотез, то поведение ступенек интересно с точки зрения выборочных свойств оценки. Удалось описать распределение длин и высот начальных и конечных ступенек; доказать, что, осуществив некоторое функциональное преобразование над оценкой, ее можно свести к пуассоновскому процессу.

Исследование свойств оценки в ситуации с трендом при асимптотическом подходе зависит от связи растущего до бесконечности объема выборки и характера изменения значений параметра. Если достаточно большому количеству наблюдений отвечает приблизительно одно и то же значение параметра, то оценка этого значения оказывается состоятельной. С практической точки зрения полученный результат означает, что если тренд несуществен, то для "средних" по порядку наблюдений оценки будут достаточно близки к априорному закону изменения. Иные свойства присущи "крайним" по порядку на-

блюдениям, им свойственно существенное искажение истинного значения параметра.

Рассмотрим теперь статистическую проблему принятия решения, когда отклонение от нулевой гипотезы об однородности выборки выражается в монотонной упорядоченности распределений ее элементов. Критерий отношения правдоподобия основывается на статистике вида:

$$S = \frac{\max\limits_{\{\vec{\lambda}|H_1\}} L(\vec{\lambda})}{\max\limits_{\{\vec{\lambda}|H_0\}} L(\vec{\lambda})}.$$

Обратимся к случаю нормального распределения и тренда среднего μ при постоянной дисперсии σ^2. Для наиболее интересного случая, когда среднее и дисперсия априори неизвестны, статистику S можно привести к виду:

$$\bar{E}^2 = \frac{\sum\limits_{i=1}^{k} m_i(\mu_i^* - \bar{\mu})^2}{\sum\limits_{i=1}^{N}(x_i - \bar{\mu})^2}, \qquad \bar{\mu} = \sum_{i=1}^{N} x_i/N$$

Основополагающим результатом является то, что условное по K распределение указанной статистики есть бета распределение с числом степеней свободы $(\frac{k-1}{2}, \frac{N-k}{2})$. Это позволяет получить общий вид интересующего нас безусловного распределения

$$Pr\{\bar{E}^2 \le u\} = \frac{1}{N} + \sum_{i=2}^{N-1} p(N,i)\, B_u\left(\frac{i-1}{2}, \frac{N-i}{2}\right), \quad 0 \le u \le 1.$$

Воспользовавшись представлением характеристической функции, можно получить предельное распределение

$$\bar{E}^2 \to N(\ell nN/N,\ 3\ell nN/N^2).$$

Применить на практике данный результат не удается, так как сходимость крайне медленная. По этой причине авторами были построены таблицы квантилей, отвечающих вероятностям 0,9; 0,95; 0,975; 0,99; 0,999; 0,9999 в диапазоне значений N от I до I000. Медленная сходимость к предельному распределению является основным препятствием для использования критерия отношения правдоподобия в случае, когда предположение о нормальности не действует, хотя асимптотическое распреде-

ление соответствующей статистики известно в достаточно общих предположениях.

Расширение полученных результатов осуществляется авторами по трем направлениям:

- повышение размерности вектора параметров, что необходимо, например, при анализе тренда как среднего, так и дисперсии нормальной последовательности;

- переход к свободным от распределения процедурам, что позволяет отказаться от предположения о нормальном распределении;

- усложнение характера тренда, что позволяет строить более адекватные практике модели.

2. Вопросы практического использования

Основной трудностью при практическом использовании изложенных выше методов является большая вычислительная сложность алгоритмов нахождения оценки тренда.

Рассмотрим сначала, какие результаты удалось получить при решении алгоритмических вопросов. Сформулированному ранее общему алгоритму нахождения оценки монотонного тренда свойственна неоднозначность; это позволяет строить варианты алгоритма и выбирать из них оптимальные по тем или иным параметрам. Например, можно построить следующий алгоритм:

1) $j = 1$;

2) найти i такое, что

$$\lambda^*_{j,i} = \min_{j \le m \le N} \lambda^*_{j,m},$$

где $\lambda^*_{j,m}$ – оценка максимального правдоподобия параметра λ при $\lambda \in \Gamma_{j,m}$;

3) положить $\lambda^*_j = \lambda^*_{j+1} = \ldots = \lambda^*_i = \lambda^*_{j,i}$;

4) положить $j = i + 1$ и перейти к 2, если $j \le N$.

Из этого алгоритма можно получить аналитическую форму оценки тренда:

$$\lambda^*_i = \max_{1 \le j \le i} \min_{j \le m \le N} \lambda^*_{j,m}, \quad i = 1, 2, \ldots, N.$$

Возвращаясь к рассмотренному примеру, проиллюстрируем процесс получения оценки. Напомнив, что в этом случае оценка $\lambda^*_{j,m}$ есть выборочное среднее по наблюдениям $x_j, x_{j+1}, \ldots, x_m$, получаем на втором шаге алгоритма:

88; I8I,5; I34; II6,5; I49,8; I54,3;

и значение i, равное 1 (минимум достигается на правом элементе). В результате имеем $\lambda^*_1 = 88$. Положив $j = 2$, повторяем вновь второй шаг алгоритма:

275; I57; I26; I65,2; I67,6;

и $i = 4$, Следовательно, $\lambda^*_2 = \lambda^*_3 = \lambda^*_4 = I26$. Еще раз повторив второй и третий шаги алгоритма, получаем в результате оценку максимального правдоподобия монотонного тренда:

88, I26, I26, I26, 230, 230.

Можно доказать, что характеристики этого алгоритма следующие: временная сложность равна $O(N \ell n N)$ в среднем при нулевой гипотезе, емкостная сложность - N.

Другой вариант алгоритма:

I) найти первую слева группу элементов, не удовлетворяющих требуемым неравенствам;

2) заменить эту группу на оценку максимального правдоподобия;

3) перейти к I, если такая группа нашлась.

Для рассматриваемого примера первый шаг алгоритма дает группу из двух наблюдений:

275, 39.

Заменив ее элементы на оценку максимального правдоподобия, получаем последовательность чисел:

88, I57, I57, 64, 283, I77.

Взвращаясь вновь к первому шагу алгоритма, находим группу из трех чисел:

I57, I57, 64.

Заменив ее элементы на оценку максимального правдоподобия, получаем последовательность чисел:

88, I26, I26, 283, I77.

Последний раз, повторив первые два шага алгоритма, приходим к прежнему результату

88, I26, I26, I26, 230, 230.

Можно доказать, что характеристики этого алгоритма следующие: временная сложность равна O(N) в любом случае, емкостная сложность - 2·N.

Построен также последовательный алгоритм, который находит оценку тренда по последовательно поступающим данным. Его характеристики: временная сложность - O(N) в любом случае, емкостная сложность - $2 \cdot \ln N$ при нулевой гипотезе. Кроме этого, последовательный алгоритм обладает важным свойством: при поступлении нового наблюдения обновляется в среднем лишь одна ступенька. Это важно при реализации алгоритма.

Рассмотрим теперь вопросы, связанные с выяснением места предлагаемых решающих процедур среди существующих. Сравнение эффективности критериев существенно упрощается при использовании асимптотического подхода. В данной работе оно проводилось с помощью асимптотической относительной эффективности по Питмену. Необходимость параметризации тренда компенсировалась широким спектром исследованных законов изменений, отвечающих различным типам взаимосвязи порядка следования и выборочных знаний. Рассматривались следующие виды тренда: $\lambda_i \sim i^{1/10}, i^{1/3}, i^{1/2}, \ln i, i, i^2, i^3, i^{10}, e^i$, модель разладки.

В рамках модели монотонного тренда используются три типа решающих процедур. Во-первых, процедуры, основывающиеся на линейной комбинации наблюдений; далее, ранговые критерии, которые составляют более представительный класс. Источником иных критериев является метод максимального правдоподобия. Сюда относятся:

- критерий числа ступенек;
- условный по числу ступенек критерий $\bar{E}^2$;
- безусловный критерий $\bar{E}^2$.

При асимптотическом подходе распределения статистик, линейно выражающихся через наблюдения, как при нулевой, так и конкурирующей гипотезах описываются с помощью специальной центральной предельной теоремы. Для ранговых статистик описать мощность соответствующего критерия в явном виде можно, если рассматривать проблему решения на континуальных альтернативах. Доказано [4], что ранговые статистики распределены асимптотически нормально как в случае нулевой, так и в случае конкурирующей гипотез, и найдено выражение для асимптотической мощности.

Сравнение критериев, свойства которых можно описать теоретически, проводилось следующим образом. Используя асимптотические распределения статистик, находилась асимптотическая относительная эффективность по Питмену всех этих критериев по отношению к наименее мощному ранговому критерию. Затем по полученным выражениям вычислялись ее значения для всех перечисленных выше видов тренда. Проведенный анализ позволил сделать следующие выводы:

- асимптотически наиболее мощные критерии (сюда относятся те, которые основываются на линейной комбинации наблюдений, и некоторые ранговые) теряют свои преимущества как только наше представление о характере тренда начинает отличаться от истинного положения дел;

- все ранговые критерии теряют в мощности с увеличением отклонения характера тренда от линейного.

Выводы относительно критериев, следующих из принципа максимального правдоподобия, были получены методом статистических испытаний. Результаты моделирования показали, что из перечисленных критериев наибольшей мощностью обладает безусловный $\bar{E}^2$, причем, нет веских оснований считать, что его мощность уменьшается при все большем отклонении от линейного тренда.

Сравнительный анализ всех критериев проводился на основании данных об асимптотической относительной эффективности критериев, основывающихся на линейной комбинации наблюдений, и ранговых критериев и данных о мощности критериев отношения правдоподобия, полученных при моделировании. Определялись объемы выборок, на которых критерии первых двух типов достигают той же мощности, что и критерии отношения правдоподобия. Затем вычислялись значения относительной эффективности для всех критериев и для всех перечисленных выше видов тренда. Полученные результаты позволили сделать вывод о целесообразности использования безусловного критерия по следующим причинам:

- с алгоритмической точки зрения он находится на уровне самых быстрых ранговых критериев;

- при отсутствии априорной информации о характере тренда его мощность в среднем выше, чем у рассмотренных критериев.

Литература

I. Тьюки Дж. Анализ результатов наблюдений. М.: Мир, 1981. - 693 с.

2. Statistical inference under order restrictions: The theory and application of isotonic regression / R.E. Barlow, D.J. Bartholomew, J.M. Bremner, H.D. Brunk. - London, New York, Sydney, Toronto: John Wiley & Sons, 1972. - 388 p.

3. Miles R.E. The complete amalgamation into blocks, by weighted means, of a finite set of real numbers // Biometrika. - 1959. - 46. - P. 317-327.

4. Гаек Я., Шидак З. Теория ранговых критериев. - М.: Наука, 1971. - 375 с.

M. Krivenko, I. Matskevich, L. Vyhandu

Statistical Analysis of a Trend

Abstract

In this paper a very quick linear algorithm to estimate monotonic trends in time series is represented. Simple estimates for statistical hypotheses of ascending, descending and no trend are given.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 683.05

П.Л. Выханду

ОБ ОДНОМ СПОСОБЕ УСКОРЕНИЯ ОБНОВЛЕНИЯ ДАННЫХ В ИНВЕРТИРОВАННЫХ ФАЙЛАХ

I. Введение

Одним из наиболее распространенных способов хранения данных являются инвертированные файлы. Основные преимущества инвертированной организации файла заключаются в быстроте поиска, достигаемой за счет ликвидации всех указателей связи в основном файле и за счет сведения большей части поиска лишь к поиску по справочнику.

Хотя инвертированная структура файла привлекательна с точки зрения быстроты поиска и получения ответа, необходимо упомянуть и ряд налагаемых ею ограничений. Во-первых, размеры справочника зачастую столь велики, что поиск по нему требует иногда значительных усилий. Во-вторых, поскольку справочник необходимо вести в определенном порядке, задачи, связанные с его ведением, весьма сложны. При добавлении нового документа требуется столько обращений к справочнику, сколько ключей присвоено этому документу; необходимо также менять все соответствующие инвертированные списки.

Во втором пункте данной статьи рассматривается, как проводится поиск в инвертированных файлах и дается оценка поиска и обновления данных. В третьем - представляется подход, где инвертированные списки предлагается заменить бинарными поисковыми деревьями, что улучшает и поиск и обновление данных. В заключении статьи приводятся выводы.

2. Поиск в инвертированных файлах

В этом пункте рассмотрим некоторые алгоритмы для поиска в инвертированных файлах [2]. Сперва введем определения, нужные для описания алгоритмов.

Список первичных (основных) ключей, имеющих общий одинаковый атрибут, называется инвертированным списком.

Запросом называется требование найти в файле группу записей, имеющих определенные свойства. Каждый атрибут, являющийся в запросе, называется операндом. Запрос состоит из операндов, между которыми производятся булевые операции & и V.

Поиск по запросу, содержащему только операторы &, производится по следующему алгоритму:

АЛГОРИТМ ANDQUERY

1. Определить каждый инвертированный список, являющийся операндом запроса.

2. Поставить пойнтер в заголовок каждого списка. Если появляется пустой список, то конец алгоритма.

3. Найти наименьшее значение, на которое указывают пойнтеры инвертированных списков. Присвоить это значение отдельному ключу, называемому активным ключом.

4. Сравнить ключ каждого списка, на который указывает пойнтер с активным ключом. Если каждый ключ соответствует активному ключу, то запись удовлетворяющая запросу, найдена.

5. Передвинуть каждый пойнтер, значение которого соответствует активному ключу, на одну позицию вперед. Если появляется пустой список, то конец алгоритма. В противном случае идти на шаг 3.

Поиск по запросу, содержащему только операторы V, производится по следующему алгоритму:

АЛГОРИТМ ORQUERY

1. Определить каждый инвертированный список, являющийся операндом запроса.

2. Поставить пойнтер в заголовок каждого списка. Если все списки пустые, то конец алгоритма.

3. Присвоить активному ключу наименьшее значение, на которое указывают пойнтеры инвертированных списков.

4. Если ключ, на который указывает пойнтер, имеет значение активного ключа, то запись удовлетворяет запросу.

5. Передвинуть каждый пойнтер, значение которого соответствует активному ключу, на одну позицию вперед. Если все списки пустые, то конец алгоритма. В противоположном случае идти на шаг 3.

Проблема усложняется, если поиск производится по составному запросу. Составной запрос - это запрос, где используют оба оператора & и V и запрос задается в виде

$$Q[(L_1 \& L_2 \& L_3) \ V \ (L_4 \& L_5)]$$

где каждый L_i представляет имя инвертированного списка. Поиск производится по следующему алгоритму:

АЛГОРИТМ COMPOUNDQUERY

I. Определить все списки, являющиеся операндами запроса.

2. Для каждого множества операндов, между которыми производят операции V и которые появляются в скобках, создать новый инвертированный список, используя алгоритм ORQUERY.

3. Используя алгоритм ANDQUERY для всех списков, оставшихся в запросе после выполнения шага 2, найти записи, удовлетворяющие запросу.

При поиске по этим алгоритмам надо просматривать все инвертированные списки, являющиеся операндами запроса. Отсюда вытекает, что время поиска пропорционально длине запроса и определяется длиной инвертированных списков справочника.

Затруднена и корректировка справочника, так как инвертированные списки справочника должны быть упорядоченными.

Кроме того из-за больших размеров справочника дополнительный расход памяти оказывается весьма значительным.

3. Использование бинарных деревьев в качестве инвертированных списков

В алгоритмах предыдущего пункта инвертированными списками являлись стеки. На этих структурах и добавление и удаление ключа в среднем требуют $O(n/2)$ перестановок, где n – число ключей в стеке вопрос. Возникает вопрос: нельзя ли заменить стеки более подходящими структурами данных, где проводить поиск и обновление данных окажется полегче.

Существует одна возможность – заменить стеки списками с одной ссылкой. Тогда алгоритмы пункта 2 останутся прежними (за исключением прохождения списка), а при обновлении данных нет надобности в перестановках, обновляются только ссылки. Однако дополнительный расход увеличивается. Кроме того, при большом объеме справочника обыкновенный последовательный поиск будет протекать крайне медленно.

С точки зрения быстрого поиска заманчиво использовать бинарные поисковые деревья, Но так как представление деревьев обычно использует две ссылки, то расход дополнительной памяти еще больше, чем при использовании списков с одной ссылкой.

Однако существует возможность представления бинарных деревьев без ссылок [I]. Полные бинарные деревья удобно хранить в последовательных ячейках памяти, т.е. в стеке, как показано на рисунке I.

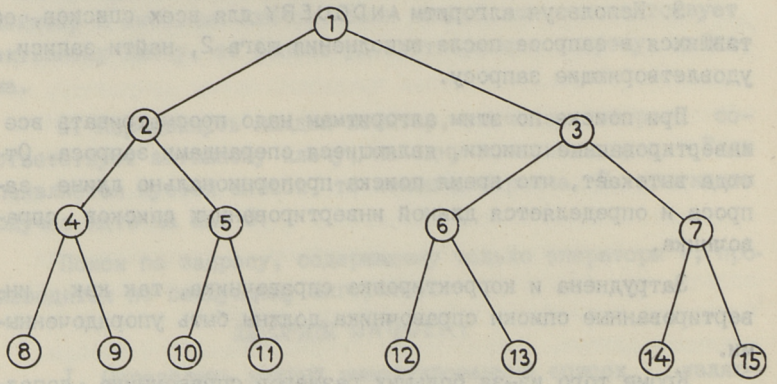


Рис. I. Последовательное распределение памяти для полного бинарного дерева.

Заметим, что отцом узла номер К является узел К/2, а его потомками являются узлы 2К и 2К+1.

Таким же образом можно представить и неполные бинарные деревья, оставляя пустые места для узлов, которых нет в дереве.

При использовании такой структуры дополнительный расход памяти по сравнению со стеками увеличивается незначительно, а все свойства бинарного дерева сохраняются.

Ниже предлагаются алгоритмы для поиска, создания и обновления бинарного поискового дерева, расположенного в массиве, и где путь его прохождения можно вычислить. Алгоритмы разработаны с использованием идей из [1, 3].

Алгоритм FORM создает глобально сбалансированное бинарное дерево в массиве BIN. В алгоритме используется массив ARRAY для упорядочения ключей так, что

ARRAY (i-1) < ARRAY (1) < ARRAY (i+1).

Для создания бинарного поискового дерева, сперва находят медиану (n+1)/2 всех ключей. Элемент на этом адресе становится корнем дерева. Остальные (n-1)/2 элементы, со значениями меньшими чем у корня, образуют левое поддерево, и другие (n-1)/2 элементы, с большими чем у корня значениями, образуют правое поддерево. Используя рекурсивно эту процедуру, получаем нужное дерево.

Для организации рекурсии используют вспомогательные массивы LOW, HIGH и MID, где запоминают порядок вставки элементов массива ARRAY в дерево. Рекурсивно используемая в алгоритме процедура описана на шагах 4–9, при этом ею управляют три условия:

1) если L>H, то имеется пустое поддерево;

2) если L=H, то имеется лист дерева;

3) если L<H, то имеется разбиваемое поддерево.

АЛГОРИТМ FORM

1. Если N<2, то конец алгоритма.

2. Сортировка массива ARRAY. Для этого подходит любой стандартный алгоритм.

3. Начальная установка L= 1, H = N, CNT=0, ADR=1 и P=1.

4. Если L>H, то перейти на шаг 8.

5. Установить M = (L+H)/2.

6. Если ARRAY(M) = BIN(P), то перейти на шаг 8.

7. Установить BIN(ADR)=ARRAY(M) и MID(ADR)=M, LOW(ADR)=L, HIGH(ADR)=H. Увеличивать CNT=CNT+1.

8. Если CNT=N, то конец алгоритма.

9. Увеличивать P = P + 1.

10. Установка левого потомка P-го узла. Для этого установить L=LOW(P), H=HIGH(P)-1, ADR=2*P и шагами 4-9 установить узел.

11. Установка правого потомка P-го узла. Для этого установить L=LOW(P)+1, H=HIGH(P), ADR=2*P+1 и шагами 4-9 установить узел.

Для поиска используется алгоритм SEARCH. Дан массив ключей BIN, образующий бинарное дерево. Производится поиск заданного ключа K.

АЛГОРИТМ SEARCH

1. Установить P = 1. Прохождение дерева начинается с корня.

2. Если BIN(P) = ^, то перейти на шаг 6.

3. Если K<BIN(P), то установить P = 2*P и перейти на шаг 2.

4. Если K>BIN(P), установить P = 2*P + 1 и перейти на шаг 2.

5. K=BIN(P), удачный поиск. Конец алгоритма.

6. Неудачный поиск. Конец алгоритма.

Для обновления данных во-первых используется алгоритм DELETE. Из массива BIN, образующего бинарное дерево, удаляется заданный ключ K. Переменная P является адресом ключа K.

АЛГОРИТМ DELETE

1. Используя алгоритм SEARCH найти удаляемый ключ.

2. Установить P = 2*P + 1 и S= 2*P. Если BIN(S)=^, то перейти на шаг 5.

3. Установить V = 2*S. Повторять шаг до тех пор, пока не получим BIN(V) = ^.

4. Присвоить BIN(P) = BIN(S) и BIN(S) = ^. Конец алгоритма.

5. Присвоить BIN(P) = BIN(R) и BIN(R) = ^. Конец алгоритма.

Для добавления используется алгоритм ADD, который в подходящее место массива BIN, образующего бинарное дерево, вставляет новый ключ K.

АЛГОРИТМ ADD

I. Установить P = 1.

2. Если BIN(P) = ^, то перейти на шаг 6.

3. Если K < BIN(P), установить P = 2*P и перейти на шаг 2.

4. Если K > BIN(P), установить P = 2*P + 1 и перейти на шаг 2.

5. Такой ключ уже имеется. Конец алгоритма.

6. Вставить ключ в дерево. Для этого присвоить BIN(P) = K. Конец алгоритма.

Поиск по запросам можно провести по алгоритмам, приведенным в пункте 2, но с той лишь разницей, что прохождение инвертированных списков производится иначе. Для прохождения всех ключей в бинарном дереве в возрастающем порядке используется алгоритм TRAVERSE.

АЛГОРИТМ TRAVERSE

I. Установить P = 1, I = 0.

2. Если BIN(P) = ^, то перейти на шаг 5.

3. Установить I = I + 1, STACK(I) = P.

4. Установить P = 2*P и перейти на шаг 2.

5. Если I = 0, то конец алгоритма.

6. Установить P = STACK(I), I = I - 1.

7. Выводить BIN(P)

8. Установить P = 2*P + 1 и перейти на шаг 2.

4. Заключение

Сперва дадим оценку скоростным свойствам инвертированных файлов, как описано в пункте 3. Поиск по запросу немного замедляется, так как прохождение инвертированных списков сложнее и требует дополнительных обращений к вспомогательным стекам. Зато значительно улучшается обновление данных.

Для локализации удаляемого ключа или нахождения подходящего места для добавления ключа в инвертированный список по описанному в пункте 2 методу в среднем понадобится $O(n/2)$ сравнений и для обновления - столько же перестановок. При замене инвертированных списков структурами, предлагаемыми в данной статье, соответствующие оценки - $O(\log_2 n)$ и $O(1)$.

Надо заметить, что при хранении бинарного дерева в массиве, индексы, указывающие на узлы дерева с ключами исходного значения, будут быстро расти. Поэтому чтобы избежать переполнения массива, надо постоянно следить за значениями индексов. Если же это все-таки произойдет, то дерево надо перебалансировать. Отсюда следует, что в данном методе предпочтительнее оперировать ключами случайного характера.

Литература

I. Кнут Д. Искусство программирования для ЭВМ. Т.3. Сортировка и поиск. - М.: Мир, 1979.

2. Johnson L.F., Cooper R.H. File techniques for data base organization in COBOL. - Prentice-Hall, 1981.

3. Chang H., Iengar S.S. Efficient algorithms to globally balance a binary search tree // Communications of the ACM. - July 1984. - Vol. 27. N 7.

P. Vyhandu

How to Quicken Data Updating in Inverted Files

Abstract

In this paper some aspects of searching and updating data in inverted files are discussed. It is suggested that binary search trees used as inverted lists, make the updating much easier. To avoid the growth of additional memory for maintaining the directory, the links are removed from the trees and traversing them is computable.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

Ю.Р. Йоонсаар, Ю.М. Ноор

ПРИМЕНЕНИЕ СПТ ПРИ ПОСТРОЕНИИ ГЕНЕРАТОРА ОТЧЕТОВ

При построении генератора отчетов систему построения трансляторов (СПТ) можно применять в принципе для реализации всех входных языков (языка описания отчетов, языка описания исходных данных и т.д.). Но здесь рассматриваются проблемы реализации только главного входного языка - языка описания отчетов. Сущность этих проблем изложена на материале реализации языка описания отчетов ОКО на базе СПТ ЭЛМА.

В первом пункте на примере создания отчета изложено описание упрощенного варианта языка ОКО.

Во втором пункте коротко описан порядок использования СПТ ЭЛМА.

В третьем пункте предлагается метод применения СПТ при реализации языка описания отчетов.

I. Краткое описание языка ОКО

Язык ОКО создан на основании промежуточного языка генератора отчетов, описанного в статье [2].

В связи с тем, что в рамках настоящей статьи нет необходимости в изложении синтаксиса языка ОКО, а требуется дать лишь общее представление о нем, то удобнее всего сделать это на примере описания отчета.

Пример.

Необходимо создать список студентов определенного факультета в разрезе учебных групп. Исходные данные в программе отчета обозначены следующими именами:

FAKUL - факультет,

GR - учебная группа,
FAM - фамилия студента,
IMJA - имя студента,
MATR - номер зачетной книжки.

Одна из возможных программ решения этой задачи выглядит следующим образом:

```
1 SECTION CNT 1;
1 P(5,3) STR FAKUL;
1 P(0,2R) STR 'ФАКУЛЬТЕТ';
1 P(2,9) STR 'СПИСОК СТУДЕНТОВ';
  2 SECTION BREAK(GR) CNT 2;
  2 P(3,12) STR 'ГРУППА';
  2 P(0,2R) STR GR;
  2 P(2,1) STR 32'-';
  2 P(1,2) STR '№';
  2 P(0,7R) STR 'ФАМИЛИЯ';
  2 P(0,9R) STR 'ИМЯ';
  2 P(0,5R) STR '№ ЗАЧ.КНИЖКИ';
  2 P(1,1) STR 32'-';
    3 SECTION BREAK(MATR) UPDATE CNT 1 CNT 2;
    3 P(1,1) CNT 2;
    3 P(0,1R) STR'.';
    3 P(0,5) STR FAM;
    3 P(0,21) STR IMJA;
    3 P(0,34) MATR;
  2 P(1,1) STR 32'-';
  1 P(2,6) STR 'ВСЕГО';
  1 P(0,2R) CNT 1;
  1 P(0,2R) STR 'СТУДЕНТОВ';
```

Отчет, создаваемый по этой программе, представлен на таблице I.

Комментарии к программе отчета:

I. Каждое предложение программы начинается с номера уровня. Программа имеет три уровня:

а) уровень факультета, значит уровень отчета в целом,
б) уровень учебной группы,
в) уровень студента (точнее номера зачетки).

Каждая часть отчета, описанная определенным уровнем программы, называется подотчетом. Благодаря введению понятия уровня программа отчета отражает структуру самого отчета.

Надо сказать, что отчет делится не только на подотчеты, но и на страницы. Следовательно, программа отчета может содержать и уровень страницы.

2. SECTION. Начальное предложение уровня.

3. BREAK. Выполнение предложений данного уровня происходит при изменении значения переменной в скобках.

4. P. Предложение вывода. Первый номер в скобках определяет ряд, а второй – позицию. Если за вторым номером следует буква "R", то позиция исчисляется не с начала ряда, а с первого свободного столбца после последнего вывода. Ключевое слово "STR" указывает на вывод символьных данных.

5. CNT. Внутренняя функция подотчета. Для их различения каждая внутренняя функция имеет еще порядковый номер.

6. UPDATE. Вычисление текущих значений для внутренних функций на данном уровне. Наличие обозначения внутренней функции без ключевого слова "UPDATE" определяет уровень получения начального значения.

Таблица I

Экономический факультет
СПИСОК СТУДЕНТОВ
ГРУППА ТР-57

№	ФАМИЛИЯ	ИМЯ	№ ЗАЧ.КНИЖКИ
1.	ИВАНОВ	СЕРГЕЙ	IIIIII
2.	ПЕТРОВА	АННА	222222
...			

ГРУППА ТР-77

№	ФАМИЛИЯ	ИМЯ	№ ЗАЧ.КНИЖКИ
1.	СИДОРОВА	ЕЛЕНА	333333
2.	ТАРАСОВ	ИВАН	444444
...			

ВСЕГО 300 СТУДЕНТОВ

2. Краткие сведения о СПТ ЭЛМА

Применение СПТ ЭЛМА можно разбить на три этапа:

1) представление синтаксиса языка;

2) представление семантики языка;

3) выполнение программы, написанной на реализованном языке.

Синтаксис языка представляется в виде грамматики предшествования, удовлетворяющей условиям, изложенным в статье [1].

С каждым правилом синтаксиса языка можно связать ряд семантических действий, написанных на языке, который в большей мере сходен с языком FORTRAN. По семантическим действиям генерируется семантическая программа на языке FORTRAN. Выполнение программы, написанной на реализованном языке начинается с синтаксического анализа, и затем программа выполняется с начала до конца с помощью заранее сгенерированной семантической программы.

3. Возможности применения СПТ при реализации языка описания отчетов

При реализации языка описания отчетов СПТ можно применять различными способами. Например, программу отчета можно выполнять, либо только до обработки исходных данных отчета, либо во время обработки этих данных.

Если к программе отчета обращаются только перед обработкой исходных данных, то во время ее выполнения необходимо построить некую подходящую промежуточную форму, на базе которой позже создается отчет. Применение такого варианта повлечет за собой довольно трудоемкое проектирование и программирование промежуточной формы. По мнению авторов, разумнее избежать использования сложной промежуточной формы (форма тем сложнее, чем мощнее язык описания отчетов) и воспользоваться вторым вариантом. С этой целью для языка описания отчетов надо дать семантику, по которой допустимо создание отчета.

Ясно, что создание отчета при однократном обращении к программе отчета невозможно. Следовательно, обращение долж-

но быть многократным, но при обработке каждой записи исходных данных надо решать, нужно ли обращение к программе отчета или нет. Перед обращением надо определить, какие именно предложения программы отчета выполнить. Для принятия этих решений нужна дополнительная информация, которую можно найти в самой программе отчета. Следовательно, для сбора и хранения этой информации надо обратиться к программе отчета еще до обработки исходных данных. В целях сокращения времени создания отчета для языка отчета разумно генерировать две семантические программы: первая - для сбора информации перед обработкой исходных данных и вторая - для создания отчета. При представлении семантических действий для второй программы надо предвидеть возможность частичного выполнения программы отчета.

Далее рассмотрим конкретный пример реализации языка описания отчетов ОКО. Чтобы сделать это на базе СПТ ЭЛМА необходимо создать грамматику предшествования и связать ее с двумя различными семантиками, с целью генерации двух семантических программ.

Задачей первой программы является сбор следующей информации:

а) сколько уровней имеется в программе;

б) при изменении какой переменной выполняются предложения определенного уровня;

в) какие предложения выполняются в начале подотчета и какие в конце;

г) какие имена переменных встречаются в программе.

Возможности второй программы следующие:

а) выполнять предложения вывода с вычислением значений выражений;

б) присваивать внутренним функциям начальные значения и вычислять их текущие значения.

В связи с тем, что СПТ ЭЛМА допускает выполнение программы отчета только с начала до конца, частичное выполнение организуется с применением индикатора. Значение индикатора определяет необходимость выполнения семантических действий, связанных с определенным правилом грамматики. К сожалению, использование такого индикатора замедляет работу генератора отчетов.

По мнению авторов, эту проблему можно решить модифицированием СПТ таким образом, чтобы сделать возможным вход-выход программы отчета с произвольной точки, но обсуждение этой задачи выходит за рамки данной статьи.

Поскольку структуры различных языков описания отчетов в принципе сходны, то можно утверждать, что предложенный в статье метод хорошо подходит для построения генераторов отчетов самого разного характера.

Литература

I. Вооглайд А.О., Лепп М.В., Лийб Д.Б. Входные языки системы ЭЛМА // Тр. Таллинск. политехн. ин-та. - 1982. - № 524. - С. 79-96.

2. Lin Sin Cho J.R. Automatic report formatting from a report specimen // The Computer Journal. - 1982. - Vol. 25, N 2. - P. 242-247.

J. Joonsaar, J. Noor

Building Report Formatter Using Compiler-Writing System

Abstract

Possibilities of using compiler-writing system for building report formatter are given. It is shown that realisation of a report language with the help of a two-step semantic program is very effective. The method has been implemented for creating the report language OKO described in this paper.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 51:801

Х.Х. Рохтла

ГЕНЕРАЦИЯ СООБЩЕНИЙ О СИНТАКСИЧЕСКИХ ОШИБКАХ

Создание для транслятора средств генерации сообщений об ошибках - задача немалой важности. Точное и продуманное сообщение поможет пользователю транслятора точнее понять суть его ошибки и в конечном счете сократить время на отладку его программы.

В первом разделе статьи дается обзор имеющихся работ об этой области. Второй раздел посвящается генерации диагностических сообщений в системе построения трансляторов ЭЛМА.

I. Каким должно быть сообщение об ошибке

До начала 80-х годов этому вопросу уделялось мало внимания. Имеется лишь одна работа того времени, которая посвящена специально диагностике ошибок [1]. В последние годы эти проблемы исследуются уже более подробно [2, 3, 4, 5]. Резюмируем некоторые важнейшие рекомендации упомянутых статей:

1) сообщения об ошибках должны быть ориентированы на пользователя, т.е. проблемы должны сообщаться в терминах, знакомых пользователю, а не в "мистических" для него внутренних терминах трансляторов;

2) тон сообщения должен быть уважительным ("пользователь - это хозяин, а транслятор - слуга") и положительным (не "нелегальная команда", а "используйте, пожалуйста, команду x, y или z");

3) предполагаемое местоположение ошибки должно быть точно локализовано (лучше использовать заметный указатель, чем вербальное объяснение);

4) символы, создавшие конфликтную ситуацию, должны быть указаны непосредственно в сообщении (лучше "THEN не может следовать за IF ", чем "нелегальная пара символов");

5) все сообщения должны генерироваться одним модулем;

6) сообщение должно быть лаконичным, но однозначно понятным и хорошо заметным;

7) лучше печатать сообщения сразу за ошибочной строкой, чем в конце листинга (хотя и вторая возможность имеет некоторые преимущества);

8) желательно печатать в конце листинга статистическую сводку об ошибках;

9) хорошо иметь две или больше степеней сообщений(для пользователей с разной степенью владения языком).

В работе [5] описывается эксперимент, где сообщения об ошибках транслятора с языка КОБОЛ были переделаны по советам такого рода. Результаты дальнейшего исправления ошибок пользователями были на 28 % лучше, чем при старых сообщениях.

Во многих работах исследуются вопросы экономии памяти при генерации большого числа разных и сложных сообщений [6, 7, 8]. Так, например, можно каждое сообщение разбивать на фразы и внутри транслятора представлять его в виде последовательности указателей на эти фразы. Таким образом, хранимые в одном экземпляре фразы можно использовать в целом ряде различных сообщений. Важность приемов такого рода проявляется в основном при создании трансляторов вручную, где каждое конкретное сообщение может быть выработано заранее.

В трансляторах, созданных с помощью СПТ, сообщения об ошибках должны в основном генерироваться автоматически. Лучшее изложение этих проблем, на наш взгляд, имеется в работе финского ученого С. Сиппу [9], где выдвинут тезис о том, что качество диагностики ошибок столь же важно, как и качество обработки ошибок.

Сообщения об ошибках делятся на декларативные и оперативные. Первые обозначают место ошибки и описывают суть ошибки с точки зрения синтаксического анализатора. Оперативное сообщение описывает предпринятое действие обработ-

ки этой ошибки. Отсутствие оперативного сообщения может оставить пользователя без возможности понимать суть последующих ошибок. Однако более трудной задачей является все же конструирование разумных декларативных сообщений. При применении анализаторов типа LL и LR эти проблемы разрешимы, так как свойство правильного префикса гарантирует разумное место для обозначения ошибки и в самом анализаторе имеется довольно точная информация об ожидаемых в конкретный момент символах. Но, например, анализатор предшествования, напротив, не дает точного места ошибки, и мало поможет при определении ожидаемых символов. Поэтому в таком случае обычно ограничиваются указанием места обнаружения анализатором ошибки и того, появление какого символа или соседство какой пары символов вызвало ошибку [10, 11].

2. Сообщения о синтаксических ошибках в системе ЭЛМА

В системе ЭЛМА имеется каскад средств обработки синтаксических ошибок (СО) [13,16]. Диагностические сообщения для всех средств из каскада генерируются единым модулем. Нужны две версии такого генератора, учитывающие разные требования для основных групп пользователей транслятора.

1) COMMON - для генерации сообщений рядовому пользователю транслятора. Ему в общем случае понятны только сообщения, говорящие на языке тех символов, которые встречаются в его программе (терминальные символы). В крайнем случае в сообщениях ему можно предложить нетерминальные символы, приведенные в описании языка его создателем. Вышесказанное порождает трудности в объяснении этому пользователю того, в чем состоит ошибка и как она была устранена, так как подобные объяснения зачастую возможны лишь в терминах применяемого синтаксического анализатора.

2) SPECIAL - для генерации специальных сообщений для пользователя - создателя транслятора. Этот пользователь должен быть знаком с общими принципами используемых методов синтаксического анализа и обработки синтаксических ошибок, а также с транслируемым языком на уровне правил подстановки. В его работе как раз необходима возможно полная информация о сути ошибки и действиях по ее устранению.

Подобная информация может предлагаться в терминах, доступных в конкретном случае.

Сообщения в версии SPECIAL содержат, во-первых, сообщения, аналогичные версии COMMON и, во-вторых, некоторые добавочные сообщения. Далее при версии SPECIAL рассматриваются только добавочные сообщения.

Далее рассмотрим виды всех генерируемых сообщений. Буквами X, Y, Z, X_i, Y_i, Z_i обозначают любые символы (терминальные и нетерминальные) языка. Они печатаются в виде, указанном в описании языка. Если какой-нибудь нетерминал не имеет такого вида (нетерминал был добавлен при генерации транслятора), то печатается NOn, где n - внутренний код этого нетерминала.

ДЕКЛАРАТИВНЫЕ СООБЩЕНИЯ являются общими для всех средств обработки СО. Они печатаются непосредственно под строкой текста, в которой была обнаружена ошибка.

Версия COMMON

\- - - - - - ? -

***** SYNTAX ERROR WAS FOUND NEAR THE MARKED SYMBOL

(вблизи указанного символа найдена синтаксическая ошибка).

Версия SPECIAL

Добавляется сообщение, имеющее три возможных вида:

***** X CAN'T BE FOLLOWED BY Y,

***** BASE $X_1,\ldots,X_N$ IS ERRONEOUS или

***** BASE $X_1,\ldots,X_N$ IS ILLEGAL IN THE CONTEXT (X,Y)

(Y не может следовать за X, основа $X_1 \ldots X_n$ - ошибочная, основа $X_1 \ldots X$ - неприменима в контексте (X,Y)).

Эти три вида сообщений соответствуют трем разным типам СО, обнаруживаемых в синтаксическом анализаторе системы ЭЛМА.

ОПЕРАТИВНЫЕ СООБЩЕНИЯ бывают разных видов для разных средств обработки СО. Они печатаются под декларативными сообщениями.

I. Простая нейтрализация [IO]

Версия COMMON

***** ANALYSIS IS CONTINUED

(Анализ продолжается).

Простота метода нейтрализации ошибки выражается и в сообщении - больше и сообщить нечего.

Версия SPECIAL

Добавляется второе сообщение:

***** WARNING : THE FOLLOWING TEXT MAY CONTAIN
UNDETECTED ERRORS

(Предупреждение: последующий текст может содержать необнаруживаемые ошибки.)

Специалисту нужно запомнить этот недостаток простой нейтрализацией СО.

II. Глобальная нейтрализация [I5, I7]

Версия COMMON

***** ERRONEOUS Ж IS IGNORED

(Ошибочная Ж игнорируется).

Ж - это название игнорируемой конструкции.

Версия SPECIAL

Добавляется сообщение:

***** ANALYSIS IS CONTINUED FROM THE MARKED
SYMBOL IN THE {SAME / LOWER} ROW
- - - - - - - - - - - - - - - ! - - - - - - - - - - - -

(Анализ продолжается с указанного символа в той же (или следующей строке).

Два вида сообщения соответствуют нахождению конца игнорируемой конструкции в строке, содержащей ошибку или в одной из следующих строк. В последнем случае печатается оперативное сообщение над этой строкой.

III. Локальная нейтрализация [I6]

Версия COMMON

Сообщение имеет один из трех видов:

* * * * * X WAS REPLACED BY Y

* * * * * X WAS DELETED или

* * * * * X WAS INSERTED

(Символ X заменялся символом Y , символ X был пропущен или символ X добавлялся).

Три вида сообщений соответствуют трем возможным действиям нейтрализации.

Версия SPECIAL

Добавляется сообщение:

* * * * * DISTANCE : D

(Расстояние - D).

D - это локальная мера различия ошибочного и исправленного текста, характеризующая качество действия нейтрализации [I8].

IУ. Адаптивная нейтрализация [I2, I4]

Версия COMMON

В этой версии сообщения аналогичны сообщениям локальной нейтрализации.

Версия SPECIAL

Добавляется сообщение одного из трех видов

* * * * * THIS RECOVERY ACTION WAS THE FIRST
CHOICE FOR THIS TYPE OF ERROR ,

* * * * * THIS RECOVERY ACTION WAS ALREDY STORED или

* * * * * THIS RECOVERY ACTION WAS THE NEXT
CHOICE FOR THIS TYPE OF ERROR

(Действие нейтрализации было первым выбором для этого типа ошибки, действие нейтрализации было заранее записано в соответствующем файле или действие нейтрализации было следующим выбором для этого типа ошибки)

Эти три вида сообщения соответствуют трем возможностям нахождения действия нейтрализации: методом локальной нейтрализации, повторением действия, найденного заранее, и методом локальной нейтрализации с блокированием некоторых действий.

Таким образом рассмотрены все сообщения системы ЭЛМА по синтаксическим ошибкам. Так как каскад средств обработки СО - открытая система, где средства на разных уровнях могут быть изменены, возможны изменения и в соответствующих сообщениях.

Литература

1. Horning J.J. What the compiler should tell the user // Compiler Construction. An Advanced cource. - 1976. - P. 525-548.

2. Brown P.J. "My system gives excellent error messages" - or does it? // Software - Practice and Experience. - 1982. - Vol. 12. - P. 91-94.

3. Brown P.J. Error messages: the neglected area of the man/machine interface // CACM. - 1983. - Vol. 26, N 4. - P. 246-249.

4. Dwyer B. A user-friendly algorithm // CACM. - 1981. - Vol. 23, N 9. - P. 556-561.

5. Schneiderman B. Designing computer system messages // CACM. - 1982. - Vol. 25, N 9. - P. 604-605.

6. Boulton P.J.P. The generation of diagnostic messages // INFOR. - 1975. - Vol. 13, N 2. - P. 135-146.

7. Hahn V.W., Athey J.G. Diagnostic messages // Software - Practice and Experience. - 1972. - Vol. 2. - P.

8. Heaps H.S., Radhakrishnaa T. Compaction of diagnostic messages for compilers // Software - Practice and Experience. - 1977.- Vol.- P. 139-144.

9. Sippu S. Syntax error handling in compilers / Report A-1981-1. - University of Helsinki, 1981. - 92 p.

10. Ripley G.D. A simple recovery-only procedure for simple precedence parsers // CACM. - 1978. - N 11. - P. 928-930.

11. R ö h r i c h J. Automatic construction of error correcting parsers / Dissertation. - Institut für Informatik, Universität Karlsruhe, 1978.

I2. Р о х т л а Х.Х., В ы х а н д у Л.К. Самоулучшающийся метод исправления синтаксических ошибок // Автоматизация производства пакетов прикладных программ (Автоматизация производства трансляторов): Тезисы докладов - Таллин. - 1980. - С. 66-69.

I3. Р о х т л а Х Х. Каскадные средства обработки синтаксических ошибок // Всесоюзная конференция по методам трансляции: Тезисы докладов. - Новосибирск. - 1981. - С. 89-91.

I4. Р о х т л а Х.Х. Стратегия самоулучшения в исправлении синтаксических ошибок // Тр. Таллинск. политехн. ин-та. - 1981. - № 511. - С. 83-89.

I5. Р о х т л а Х.Х. Нейтрализация синтаксических ошибок в системе построения трансляторов ТПИ // Тр. Таллинск. политехн. ин-та. - 1982. - № 524. - С. 119-129.

I6. Р о х т л а Х.Х. Опыт внедрения средств обработки синтаксических ошибок // Тр. Таллинск. политехн. ин-та. - 1983, № 554. - С. 71-83.

I7. Р о х т л а Х.Х. Восстановление синтаксического анализа с помощью синхротроек // Автоматизация производства пакетов прикладных программ и трансляторов. П Всесоюзная конференция: Тезисы докладов. - Таллин. - 1983. - С. 144-146.

I8. Р о х т л а Х.Х. Об одном методе определения схожести цепочек символов // Тр. Таллинск. политехн. ин-та. - 1984. - № 568. - С. 71-80.

H. Rohtla

The Generation of Syntax Error Messages

Abstract

In the first part of this paper an overview of the work about diagnostic messages is given. The second part of the paper deals with the design of syntax error messages in the compiler writing system ELMA.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.I42

М.В. Лепп, М.А. Хейнсоо

ОПРЕДЕЛЕНИЕ ТРАНСЛЯЦИИ АДА-ДИАНА С ПОМОЩЬЮ S-АТРИБУТНЫХ ГРАММАТИК

Введение

При описании языков программирования атрибутными грамматиками более всего пользуются для представления синтаксиса и контекстных условий языков. В рамках систем построения трансляторов (СПТ), базирующихся на атрибутных грамматиках, из соответствующего описания генерируются синтаксический и семантический анализаторы (называемые FRONT END компилятора). Входом семантического анализатора, представляющего из себя вычислитель атрибутов, является дерево вывода (дерево конкретного синтаксиса). В отличие от этого в проекте компилятора языка АДА [I] входом семантического анализатора предусмотрено дерево абстрактного синтаксиса (ДАС) АДА-программы. Применение атрибутного метода для описания семантики при таком условии подразумевает возможность сопоставления семантических атрибутов и действий с описанием абстрактного синтаксиса языка. В рамках системы GAG [2] данная проблема решается очень просто - абстрактный синтаксис представляется в виде КС-грамматики, определяющей деревья абстрактного синтаксиса последовательностями вершин, соответствующими прохождению деревьев в прямом порядке.

В связи с определением входа семантического анализатора в виде дерева абстрактного синтаксиса программы, возникает проблема формального описания трансляции исходных АДА-программ в деревья абстрактного синтаксиса. Эту проблему можно решить также при помощи атрибутных грамматик. В данной статье для этой цели используются S-атрибутные грамматики. Последние являются математической моделью описания языков программирования в четвертой версии СПТ ЭЛМА [5] В системе имеются как фортрановская, так и паскалевская реализация модели.

В настоящей статье для решения проблемы - описания трансляции - используется паскалевская реализация как наиболее подходящая.

I. Внутренний язык ДИАНА компилятора АДА

При определении структуры компилятора языка АДА общепринято его расчленять на две части - Front End и Back End, где Front End осуществляет лексический, синтаксический и семантический анализы АДА-программ, а Back End - генерацию кода. Для фиксации результатов работы Front End разработаны внутренние языки, одним из которых и является ДИАНА [1]. Язык ДИАНА рассматривается как абстрактный тип данных, где объектом типа является внутреннее представление АДА-программы, а математической моделью объекта - атрибутированное дерево. Объект, принадлежащий языку ДИАНА, называется ДИАНА-деревом.

Каждая вершина дерева обладает именем, а атрибуты несут информацию, получаемую в результате работы лексического анализатора (лексические атрибуты) и семантического анализатора (семантические атрибуты). В принципе можно задавать формальное описание трансляции исходных АДА-программ непосредственно в ДИАНА-деревья. Однако, следуя отчетам [1 и 2], разумно выделить еще одну форму внутреннего представления АДА-программ - это деревья абстрактного синтаксиса (ДАС) [3]. В связи с этим соответствующая трансляция определяется в виде двух последовательных этапов - трансляция исходной АДА-программы в ДАС и трансляция ДАС в ДИАНА-дерево.

ДИАНА-дерево от дерева абстрактного синтаксиса (согласно [2] - синтаксическое ДИАНА-дерево (СДИАНА-дерево)) отличается тем, что к нему добавлены семантические атрибуты и из него устранены синтаксические неоднозначности. Синтаксическое ДИАНА-дерево является результатом работы синтаксического анализатора.

Поскольку ДИАНА является абстрактным типом данных, то помимо определения набора объектов, задаются и операции для манипуляции объектами. В основном эти операции распределяются на три класса:

определяющий тип - выдает тип вершины;
конструкторы - строят вершины и устанавливают значения атрибутов;
селекторы - обеспечивают доступ к вершинам и атрибутам.

2. Реализация ДИАНА как абстрактного типа данных на языке ПАСКАЛЬ

Реализация ДИАНА как абстрактного типа данных на некотором языке программирования состоит в выборе способа представления объектов и в реализации операций в виде процедур и функции.

Поскольку дерево во время синтаксического анализа является динамически конструируемым объектом, то на языке ПАСКАЛЬ целесообразно реализовать его через указатели. Таким образом, вершины описываются как записи с вариантами (для разных множеств атрибутов). Для спецификации вариантов используются имена вершин, в связи с чем определяется и соответствующий тип перечисления.

Для интересующей нас трансляции достаточно операций типа конструкторов - конструкторы вершин и конструкторы лексических атрибутов.

Конструкторы вершин

Функция MAKE$_i$ - для $i = 0,1,2,3$ каждая такая функция по имени строит новую вершину с i сыновьями, которые вместе с именем служат аргументами функций;

Функция LIST - строит по имени вершину специального типа, называемого в [1] вершиной типа последовательность (sequence);

Процедуры HEAD и TAIL - позволяют прикреплять сыновей к вершинам типа последовательность соответственно в начало и конец имеющейся последовательности.

В силу ссылочной связи вершин дерева, сыновья в функциях MAKE$_i$ задаются при помощи указателей. Результаты функций MAKE$_i$ и LIST также представлены указателями.

Конструкторы лексических атрибутов

Лексические атрибуты внутреннего языка ДИАНА содержат информацию о лексемах исходного текста. Такими атрибутами являются, например, сама лексема в символьном представлении и ее позиция в исходном тексте.

Процедура PUTLEX помещает в вершину значение лексического атрибута, которым является символьное представление лексемы.

3. Описание трансляции исходных программ в разреженные деревья

Перед изложением S-атрибутных грамматик для интересующей нас цели посмотрим один простой тип грамматик, позволяющих описать трансляцию исходных программ в разреженные деревья и причины, почему их недостаточно для определения трансляции АДА-программ в СДИАНА-деревья.

Разреженное дерево [4] является наиболее простым типом ДАС. Это дерево непосредственно выводимо из дерева разбора программы удалением ненужных вершин и помечиванием оставшихся вершин сходными именами, имеющими семантическое значение.

Описание трансляции исходных программ в разреженные деревья в системе ЭЛМА задается следующим образом. С каждым правилом синтаксиса, которому соответствует вершина в разреженном дереве, связывается имя (метка) конструируемой вершины. Аналогично связываются имена и с терминальными символами, которые включаются в разреженное дерево.

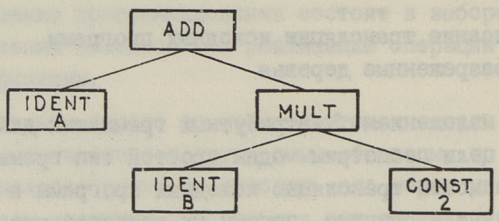
Пример.

Описание простого арифметического выражения и его разреженного дерева в системе ЭЛМА.

```
        FIXLEXEMS
IDENT :: IDENT
CONST :: CONST
    RULES
w1  E ⇒ E+T :: ADD
w2  E ⇒ T
```

```
w 3   T ⇒ T * F :: MULT
w 4   T ⇒ F
w 5   F ⇒ IDENT
w 6   F ⇒ CONST
w 7   F ⇒ (E)
```

Разреженное дерево выражения A+B*2 выглядит следующим образом:



Грамматику такого типа в [6] называют транслирующей. На уровне описания трансляции связанные с правилами имена вершин можно трактовать как имена процедур-действий, строящих соответствующие вершины. Для реализации такой транслирующей грамматики достаточно одной процедуры, которая управляется синтаксичесим анализатором и которая оперирует с указателями вершин через синтаксический магазин.

Надо отметить, что структурные связи в разреженном дереве те же, что и в дереве разбора, из которого оно выводилось. То есть, если $a_1, a_2, \ldots, a_n$ - вершины в прямом порядке дерева разбора, $b_1, b_2, \ldots, b_m$ - вершины в прямом порядке разреженного дерева и b_k соответствует a_i и b_ℓ a_j причем, если a_i предшествует a_j в последовательности $a_1, \ldots, a_n$, то b_k предшествует b_ℓ в последовательности $b_1, \ldots, b_m$.

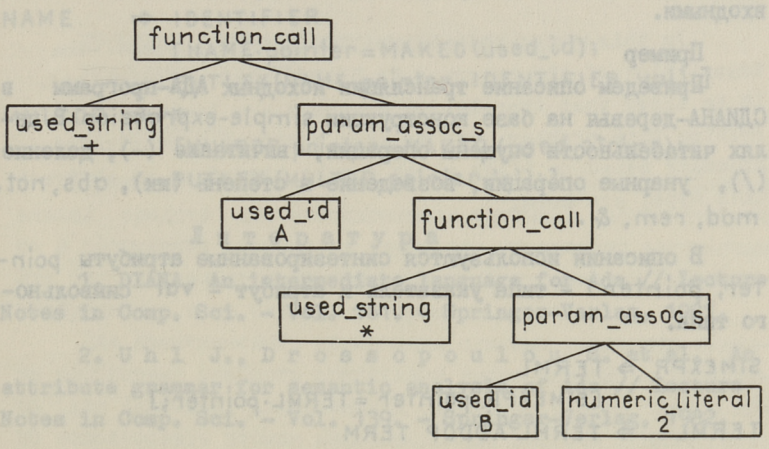
Возможность представления дерева абстрактного синтаксиса разреженным деревом во многих языках вполне удовлетворяет условиям семантического анализа.

Но разреженного дерева не достаточно, если в дереве абстрактного синтаксиса предусматриваются переструктурирование отношений, добавление вершин или семантическая унификация разных языковых конструкций. Эти действия как раз надо производить в случае языка АДА.

Например, большинство операций, появляющихся в выражениях, (в конструкциях simple-expression) на уровне абстрактного синтаксиса, следует интерпретировать как обращения к функциям. В этом случае синтаксическое ДИАНА-дерево выражения

A + B* 2

выглядит следующим образом:



S - атрибутные грамматики являются подходящим и достаточным расширением рассмотренных выше грамматик.

4. Описание трансляции АДА-программ в СДИАНА-деревья с помощью S-атрибутных грамматик

При изложении материала в настоящем пункте предполагается, что читатель хорошо знаком с понятием атрибутных грамматик.

В S-атрибутных грамматиках нетерминальные символы имеют только синтезированные атрибуты. Семантические действия соответственно имеют форму функции, где вычислимыми являются атрибуты символа левой части продукции, а аргументами - атрибуты того же символа и атрибуты символов правой части продукции.

Применение S-атрибутных грамматик при определении трансляции исходных текстов в СДИАНА-деревья базируется на использовании перечисленных в пункте 3 конструкторов в ка-

честве семантических действий, и атрибутов типа указатель, отсылающих к вершинам. Из последнего следует, что в качестве семантических действий кроме функции выступают процедуры, что не допускается в стандартном случае S-атрибутных грамматик. Однако стоит отметить, что модификация S-атрибутных грамматик таким образом не противоречит общему принципу атрибутной техники, так как все параметры процедур являются входными.

Пример

Приведем описание трансляции исходных АДА-программ в СДИАНА-деревья на базе конструкции simple-expression. В целях читабельности опущены операции, вычитание (-), деление (/), унарные операции, возведение в степень (**), abs, not, mod, rem, &.

В описании используются синтезированные атрибуты pointer, pointer1 - типа указатель и атрибут - val символьного типа.

```
SIMEXPR ⇒ TERML
          [SIMEXPR.pointer = TERML.pointer;]
TERML   ⇒ TERML ADDOP TERM
          [TERML.pointer1 = LIST(param_assoc_s);
          HEAD(TERML.pointer1, TERML.pointer);
          TAIL(TERML.pointer1, TERM.pointer);
          TERML.pointer = MAKE2(function_call, ADDOP.
                                pointer, TERML.pointer1);]
TERML   ⇒ TERM
          [TERML.pointer = TERM.pointer;]
ADDOP   ⇒ +
          [ADDOP.pointer = MAKE0(used_string);
          PUTLEX(ADDOP.pointer, '+');]
TERM    ⇒ FACT
          [TERM.pointer = FACT.pointer;]
TERM    ⇒ TERM MULTOP FACT
          [TERM.pointer1 = LIST(param_assoc_s);
          HEAD(TERM.pointer1, TERM.pointer);
          TAIL(TERM.pointer1, FACT.pointer);
          TERM.pointer = MAKE2(function_call, MULTOP.
                               pointer, TERM.pointer1);]
```

```
FACT    ⇒ PRIM
          [FACT.pointer = PRIM.pointer;]
PRIM    ⇒ CONSTANT
          [PRIM.pointer = MAKEO(numeric_literal);
          PUTLEX(PRIM.pointer, CONSTANT.val);]
PRIM    ⇒ NAME
          [PRIM.pointer = NAME.pointer;]
NAME    ⇒ IDENTIFIER
          [NAME.pointer = MAKEO(used_id);
          PUTLEX(NAME.pointer, IDENTIFIER.val);]
MULTOP  ⇒ *
          [MULTOP.pointer = MAKEO(used_string);
          PUTLEX(MULTOP.pointer,'*');]
```

Литература

1. DIANA, An intermediate language for Ada // Lecture Notes in Comp. Sci. - Vol. 161. - Springer-Verlag, 1981.

2. Uhl J., Drossopoulou S. et al. An attribute grammar for semantic analysis of Ada // Lecture Notes in Comp. Sci. - Vol. 139. - Springer-Verlag, 1982.

3. Formal definition of the Ada programming language // Honeywell Inc., Cii Honeywell-Bull, INRIA, Nov. 1980.

4. Вооглайд А.О. Семантическое равенство распознавателей, работающих на грамматике LR(к) и грамматике предшествования с (I/I) ограниченным каноническим контекстом // Тр. Таллинск. политехн. ин-та. - 1976. - № 411. - С. 39-55.

5. Вооглайд А.О., Лепп М.В., Лийб Д.Б. Входные языки системы ЭЛМА // Тр. Таллинск. политехн. ин-та - 1982. - № 524. - С. 79-96.

6. Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. - М.: Мир, 1979.

M. Lepp, M. Heinsoo

Definition of ADA - DIANA Translation Using S-Attributed Grammars

Abstract

In this paper the possibilities to describe the translation of ADA-programs to DIANA-trees are examined. It is shown that translating grammars are insufficient for solving this problem. Due to that modified S-attributed grammars based on DIANA which can be viewed as an abstract data type, are used.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.06

Т.Й. Микли

ПРОБЛЕМЫ СОЗДАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

I. Введение

Прикладные системы реального времени являются одним из важных средств для автоматизации рутинной работы при сборе и оформлении данных, а также для управления автоматами. Эти системы зачастую применяются на предприятиях бытового обслуживания и на складах материалов и т.п.

Для практического применения системы реального времени необходимо снабдить средствами информационного характера (для запросов, для создания отчетов). В этом случае можно использовать термин информационные системы реального времени (ИСРВ). Более широкое применение ИСРВ на практике начнется с появлением микроЭВМ. В настоящее время в роли главного технического средства для ИСРВ выступает локальная сеть, составленная из микрокомпьютеров.

В статье предлагается подход, по которому ИСРВ создается с помощью системы обработки данных, состоящей из пакетов (подсистем), способной настраиваться на многие приложения. В таком случае необходимо определить состав и функциональные свойства этих пакетов - подсистем. Для этого необходимо исследовать сущность и технологию работы ИСРВ, а также функциональные свойства обработки данных и системные средства, необходимые для выполнения функции системы обработки данных.

2. Информационная система реального времени

Под информационной системой понимается автоматическая система сборки, ввода, хранения, выдачи и распределения информации, см. [3]. Под ИСРВ понимается такая система, в которой сборка и ввод информации приводится с некоторой точностью места и времени возникновения информации.

В основном она возникает в связи с выполнением традиционных деловых функций работника. В соответствии с реальными событиями окружающей среды (часто в соответствии с некоторой технологией, т.н. внешней технологией) на дисплее оформляются сообщения, в частности в виде документов, заказов и ордеров. Сообщения выступают в качестве данных реального времени. Хранимые данные для счета (выдачи отчетов, ответов на запросы) и распределения генерируется ИСРВ автоматически.

В соответствии с внешней технологией информация окружающей среды возникает в нескольких физических местах при участии многих работников. При этом необходимо, чтобы сообщения обрабатывались и передавались другому работнику, который также участвует в этой внешней технологии. Чем быстрее эта передача производится, тем скорее можно реагировать и продолжать технологический процесс. В ИСРВ необходимо предусмотреть передачу данных (сообщений) другому пользователю или для общего пользования в базу.

Таким образом данные реального времени всегда отражают состояние окружающей среды и их можно использовать в качестве оперативного управления технологией.

Из вышесказанного следует ряд свойств для ИСРВ, которые отчасти пересекаются со свойствами, предлагаемыми Блэкманом см. [I]:

- непосредственный контакт между пользователем и ИСРВ;
- немедленная обработка данных системой;
- обслуживание одновременно нескольких пользователей;
- непредсказуемость потребности обслуживания;
- использование локальной сети микрокомпьютеров.

Необходимо отметить, что ИСРВ, которую применяют вместе с некоторой внешней технологией, определяет также свою технологию и организацию труда работников.

2.I. Непосредственный контакт

Непосредственный контакт между пользователем и системой осуществляется на рабочих местах. На микрокомпьютерах в системе необходимо предусмотреть несколько различных мест в соответствии с организацией труда внешней технологии. При создании рабочего места в ИСРВ следует учитывать, что для выполнения работ рутинного характера используются работники низкой квалификации. Это значит, что средства для этого

рабочего места должны удовлетворять специальным требованиям соответственно квалификации работника. Рабочее место имеет свое меню возможных действий пользователя, из которых каждое состоит из шагов, по которым работник вводит сообщения или параметры по внешней технологии или заполняет документы при помощи макетов таких форм.

Для дополнительного контроля пользователь может делать запросы реального времени, т.е. данных, которые он сам оформил. Последовательность шагов технологии обработки данных зафиксирована соотношением системы и диалог между системой и пользователем требует обыкновенно ответа ДА / / НЕТ.

Система должна учить пользователя исправлять обнаруживаемые ошибки так, чтобы он мог продолжать работы.

2.2. Немедленная обработка данных

Под немедленной обработкой данных понимается то обстоятельство, что не существует "большого" промежутка времени между реальными событиями и сообщениями информационной системы. Данные ИСРВ с большой точностью отражают состояние внешней технологии, но это свойство ИСРВ требует от внешней технологии, чтобы она управляла ресурсами и возможными действиями работников.

Сообщение окружающей среды принимается ИСРВ немедленно, контролируются возможности внешней технологии, оформляются новые сообщения для других рабочих мест, образуют базу данных для данных реального времени и в соответствии с действиями пользователя образуются данные для счета.

2.3. Обслуживание одновременно нескольких пользователей

Обслуживание одновременно нескольких пользователей в системах реального времени означает, что в ИСРВ моделируется организация труда внешней технологии. Работники с разными функциями должны иметь рабочее место соответственно должности.

В ИСРВ необходимо предусмотреть, во-первых, автоматическую транспортировку сообщений на другое рабочее место

(обмены сообщений); во-вторых, синхронизацию действий разных работников; в-третьих, организацию единой базы данных с данными реального времени и данными счета.

База данных обновляется многими работниками в соответствии с технологией ИСРВ. Для того, чтобы избежать намеренного искажения данных, необходимо чтобы, во-первых, работник одного рабочего места имел бы возможность выполнять только необходимые для его работы действия; во-вторых, каждое рабочее место должно иметь действие предъявления пароля.

Для функционирования ИСРВ нужны средства для работы администратора системы. Администратор поддерживает систему в работоспособном состоянии по текущему контролю функционирования, повторному запуску, восстановлению, резервированию и обеспечению целостности.

Основным техническим средством реализации ИСРВ на ЭВМ является локальная сеть микрокомпьютеров, имеющая для рабочих мест общую внешнюю память на дисках с высокой скоростью. При выборе конфигурации такой сети следует учесть нужды рабочих мест, необходимость дополнительных микроЭВМ, печатающих устройств, устройств внешней памяти для замены их при отказе. Открытые структуры локальных сетей позволяют при необходимости добавлять новые рабочие места. Кроме того, локальные сети позволяют организовать базу данных и автоматическую транспортировку данных (обмены сообщениями) между рабочими местами на общем внешнем устройстве.

3. Функции программного обеспечения ИСРВ

Функции программного обеспечения ИСРВ для системы реального времени на больших ЭВМ, предложенные Блэкманом (см. [1]), взяты за основы и дополнены для ИСРВ такими функциями, как управление сетью, управление действиями пользователя, управления технологией и т.д. Ряд функций имеют другое значение: (управление обменами, система сбора данных) и некоторые из них нужны в интересах достижения настраиваемости ИСРВ на различные приложения или для перенастраивания системы при изменении (генерация макетов: описание базы данных, описание форм сообщений).

Функции обработки данных следующие:

- управление сетью;
- управление данными (сообщениями);
- управление базами данных;
- управление обменом данных;
- закрытие, восстановление и повторный пуск;
- управление сбором данных;
- управление макетами;
- запросная система;
- генерация отчетов;
- генерация диалога;
- управление действиями пользователя;
- управление технологией информационной обработки.

Многие из этих функций можно реализовать системными средствами локальных сетей. Управление сетью, данными и обменами являются более распространенными возможностями программного обеспечения системного уровня. Другие - средствами управления базами данных, имеющими многопользовательский режим. Организацию сообщений, запросную систему и выдачу отчетов можно организовать также с помощью баз данных прикладного уровня. Закрытие, восстановление, повторный пуск требуют уже прикладного уровня. Запросная система также требует прикладного уровня. Эти функции на запросы определения состояния системы (запросы данных реального времени) и на запросы данных счета. Первая из этих функций нуждается в быстром доступе к базе данных, что, в свою очередь, требует дополнительной разработки.

Такие функции, как генерация отчетов, генерация диалога необходимо создавать в интересах гибкости и возможности перенастраивания системы.

Особую роль играют такие функции, как управление действиями пользователя и управление технологией информационной обработки. Как уже отмечалось, пользователю для рутинных работ необходимо предусмотреть параллельно возможные и повторяющиеся шаги. Возможность этих шагов зависит от внешней технологии и организации труда в ИСРВ.

Моделирование шагов действия представляет собой определенную трудность, так как их можно предсказать. Программное обеспечение для представления таких моделей должно быть гибким и оперативно реагировать на изменения.

Идея индустриального производства (см. [3]) информационных систем (в данном случае ИСРВ) требует, чтобы в качестве программного обеспечения этих систем использовалась система обработки данных, состоящая из нескольких пакетов (или систем общего назначения), которые можно настраивать на различные применения. Один из возможных способов настраивания этих пакетов при помощи языков специального назначения. Такой подход используется, например, в работах [3, 4].

4. Система обработки данных для ИСРВ

Система обработки данных для создания ИСРВ состоит из шести подсистем. Далее определяются их свойства и функции обработки данных для этих подсистем:

- подсистема управления базами данных;
- подсистема генерации и управления макетами;
- подсистема управления технологией обработки;
- диалоговый процессор;
- подсистема управления обменами данных.

4.1. Подсистема управления базами данных

Подсистема управления базами данных выполняет функцию организации базы, управление данными реального времени (сообщения) и данными для учета. Кроме того она организует в качестве данных запросы, диалоги, макеты сообщений и выполняет сборку данных, запросы данных, генерацию отчетов. База данных такого рода кроме проблемных файлов имеет системные файлы для хранения текстов запросов пользователя и диалогов, а также макеты сообщений. Свойства ИСРВ требуют, чтобы система базы данных имела многопользовательский режим на уровне записей одного файла, а также методы организации данных типа FIFO или LIFO и, наконец, для поиска записей должен быть обеспечен прямой доступ к данным.

Для описания базы данных необходимо иметь язык описания данных. язык запросов высокого уровня и язык описания отчетов.

4.2. Подсистема генерации и управления макетами

Подсистема выполняет описания макетов, свойств их полей (для контроля данных) - автоматически обновляемых данных и заполняемых пользователем данных) и интерфейса с базами данных. Макет такого рода должен иметь свойства, отвечающие эргономическим принципам, т.е. сообщения об ошибках и обучение их исправлению, видеотехнические способы для облегчения работы. По описанию генерируется программа для обновления файлов реального времени.

4.3. Подсистема управления технологией

Функциями обработки данных подсистемы управления технологией в данном случае являются:

- управление библиотеками программ;
- управление действиями пользователя;
- управление технологией информационной обработки.

Прикладной уровень этих функций состоит в том,что соответственно шагам действий пользователя-работника (точнее в соответствии с сообщениями о событии) запуск программы контролирует условия для выполнения этого шага, организует диалог и при необходимости обмен сообщениями на этом шаге, наконец, определяет последующие шаги.

Подсистема управления технологией состоит из языка описания технологии, при помощи которого описывается модель технологии, и программы, которая в соответствии с описанием выполняет информационную обработку совместно с действиями пользователя-работника.

Модель технологии состоит из действий, шагов действий, условий для шага, названий программ и названий диалога. Необходимость пуска различных программ непредсказуема. Поэтому они должны обладать свойством абсолютности, т.е. работать на многих заданиях

4.4. Подсистема управления диалогами

Подсистема организует архив из именуемых диалогов, состоящих из текстов диалога и описаний параметров. По данному проекту подсистема управления диалогами организована при помощи системы базы данных. Возможно обновление диалогов.

Подсистема включает и программные средства для пуска диалога и применения значений параметров, которые используются подсистемой управления технологией.

4.5. Подсистема управления обменами данных

Подсистема организует архив форм сообщений (макеты), передаваемых на другое рабочее место. В функции этой подсистемы входит обслуживание (выполнение функции обновления) архива, автоматическая генерация сообщений и оформление протоколов для передачи данных. Передача данных предусматривается с помощью системных средств.

Архив форм сообщений организуется средствами системы базы данных, а управление передачами сообщений на прикладном уровне производится при помощи средств подсистемы технологии. Описание данных, которые формируются в сообщении, определяется при описании форм.

5. Заключение

Принципы построения программного обеспечения, описанные в данной статье, позволяют применять систему обработки данных в нескольких прикладных заданиях, а что главное, программное обеспечение позволяет перенастраивать ИСРВ при отказе.

Литература

I. Блэкман М., Проектирование систем реального времени. - М.: Мир, 1977.

2. Якубайтис Е.А. Архитектура вычислительных сетей. - М.: Статистика, 1980.

3. Жеребин В.М., Мальцев В.Н., Соваллов М.С. Экономические информационные системы. - М.: Наука, 1978.

4. Лучковский Т.Ф., Микли Т.Й., Рензер А.В. Экономические информационные системы коллективного пользования // Тр. Таллинск. политехн. ин-та. - 1982. - № 524.

T. Mikli

Problems of Creating Real-Time Information Systems

Abstract

In this paper the contents, structure and properties of the database and software of real-time information systems are observed. The functional properties of application packages are developed and implemented.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

Л. Н. Эльмик, М.Х. Роост

БАЗА ДАННЫХ ДЛЯ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

I. Введение

Современная техника позволяет передавать большие информационные потоки с высокой скоростью и на достаточно большие расстояния. Обладание подробной, точной и своевременной информацией позволяет выйти человечеству на более высокий уровень в своей деятельности. Таким образом, центр тяжести в обработке информации переносится на создание систем реального времени.

В настоящей статье предпринята попытка обобщить некоторые аспекты системы реального времени, с которыми мы столкнулись в процессе создания информационной системы для станции технического обслуживания.

Система создавалась на ЭВМ "Искра 226" со следующими периферическими устройствами:

- терминалы VDT-52100 VIDEOTON (2 штуки)
- накопитель на гибких магнитных дисках (ИСКРА 005-50 или ИСКРА 005-51)
- накопитель на магнитных дисках (ИСКРА 005-71)
- накопитель на магнитной ленте (ИСКРА 005-60)
- печатающее устройство ("ROBOTRON 1154" или "DZM-180")

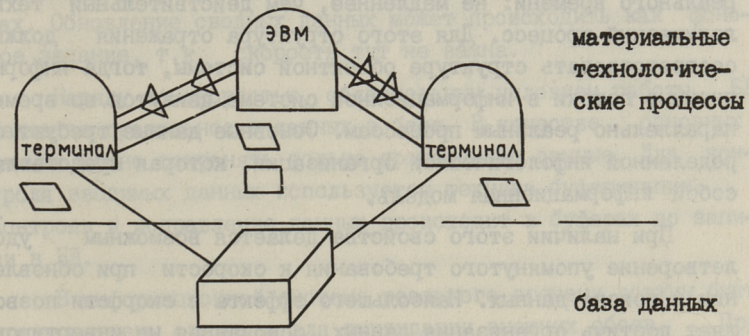
В качестве операционной системы использовалась система "Скоропись", разработанная в ВНИКИ СЧПУ ЛЭМЗ, с входным языком "Скоропись", на котором и создавалось все программное обеспечение системы.

Основной задачей было создание системы, работающей в реальном времени. Такая система, по Мартину [1], может быть охарактеризована как система, "которая управляет внешними объектами, получая информацию, обрабатывая ее и возвращая результаты достаточно быстро для того, чтобы воздействовать

на функционирование внешних объектов в почти тот же момент времени". Пользователями системы реального времени могут быть как люди, так и приборы. В нашем случае пользователями системы являлись администрация станции обслуживания, цеховой мастер, кладовщик и администратор информационной системы.

Основная идея проекта заключалась в том, чтобы информационные потоки двигались параллельно по времени с технологическими процессами. (В отличие от пакетных систем, где информационная система отражает действительное положение вещей с существенной удержкой)

Наглядно систему можно изобразить следующим образом:



Реализуемая информационная система состоит из следующих пакетов:

- подсистема управления технологии,
- подсистема технологического макетного обмена,
- подсистема управления файлами,
- подсистема вывода (языка запросов, технологический запрос, отчет).

В данной статье рассматривается два аспекта создаваемой информационной системы: требования к базе данных реального времени, средства и способы их выполнения; особенности использования базы данных реального времени.

2. Требования к базе данных

При создании базы данных (БД) реального времени особого внимания требуют:

- достоверность вводимых данных и их защита при высокой скорости передачи и обработки данных,
- быстрое восстановление базы данных,
- одновременный доступ к данным для многих пользователей,
- непредсказуемость потока обращений к базе данных [4]

Чтобы информационная система работала в реальном времени ей надо работать со скоростью, отвечающей требованиям реального времени: не медленнее, чем действительный технологический процесс. Для этого структура отражения должна соответствовать структуре объектной системы, тогда информационные потоки в информационной системе движутся во времени параллельно реальным процессам. Основные данные требуют определенной инфологической организации, которая представляет собой информационная модель.

При наличии этого свойства делается возможным удовлетворение упомянутого требования к скорости при обновлении и поиске данных. Наибольшего эффекта в скорости позволяет достичь организация данных, основанная на инвертированных файлах.

Ввиду невозможности предусмотреть все возможные обращения к базе данных, следует использовать прямой доступ по многим ключам для поиска записей. И этому требованию тоже удовлетворяет база данных, основанная на инвертированных файлах. Таким образом, информационная модель и необходимость поиска по многим ключам приводят к использованию инвертированных файлов.

Обработка данных в масштабе реального времени предполагает практически одновременный доступ к данным для многих пользователей. Существенным при этом является сохранение целостности БД [5]. Технологические процессы, которые обновляют одни и те же записи базы, не должны мешать друг другу. Следует исключить возможность чтения противоречивых данных. Последние могут появиться в результате неоконченного обновления. Возникновение противоречивых данных

исключается путем введения замков и протоколов [2]. Проблема решается путем использования многопользовательского режима на уровне записей одного файла. Этого требует скорость поиска в базе данных.

Процесс обновления записи распадается на две фазы. На первой фазе происходит чтение записи в буфер, затем ее обновление. На второй фазе обновленная запись пишется обратно в файл. На обеих фазах запись защищена от всех остальных обновлений. На второй фазе запись защищена и от чтения тоже.

В интересах увеличения скорости поиска, а также упрощения алгоритмов представляется целесообразным хранить данные реального времени отдельно от сводных данных в разных файлах. Обновление сводных данных может происходить как фоновое задание, т.к. скорость тут не важна.

Наряду со скоростью, обязательным условием работы БД является правильность данных в базе. В качестве основных данных можно сохранять только проверенные данные. Для контроля вводимых данных используется техника буферизации. Контроль и исправление данных происходит в буферах до записи в БД.

В информационной системе реального времени должны быть предусмотрены средства для ликвидации внешних сбоев. При возникновении необходимости восстановления БД, следует обеспечить средства восстановления состояния БД на момент сбоя. Для этого надо постоянно вести протокол работы БД, а также время от времени копировать все файлы БД. Это одновременно служит защитой от порчи данных.

Функции и методы защиты данных сильно переплетены между собой. Для защиты данных от несанкционированного доступа на физическом уровне их следует копировать, а на системном - использовать коды защиты. Такая защита может быть как на уровне файлов, так и на уровне полей данных. Код защиты файла является более общим, код защиты поля - более детальным.

3. Особенности ввода-вывода в информационной системе реального времени

В общем виде ввод-вывод должен удовлетворять определенным требованиям, которые условно можно подразделить на две группы: эргономические и технологические.

В аспекте эргономики ввод-вывод должен обеспечивать простой, гибкий, полный и произвольный доступ к базе данных в период создания технологического процесса. Во время эксплуатации информационной системы такой доступ должен иметь администратор (лицо, ответственное за информационную систему).

В аспекте технологии ввод-вывод, связанный с технологическим процессом, должен удовлетворять следующим требованиям:

I) Ввод-вывод в каждом узле технологического процесса должен

- обеспечивать работающего в узле человека всей необходимой ему информацией из базы данных;
- вносить изменения в базу данных, отражающие обновление информации в узле.

2) Ввод-вывод должен выдавать информацию о результатах технологического процесса и о ресурсах (выполнен ли тот или иной заказ? какие работы еще предусмотрены? какие детали имеются на складе? и т.д.).

3) Ввод-вывод должен выдавать сводную информацию о технологическом процессе за некоторый промежуток времени.

Выполнение этих технологических требований приводит к определенным результатам.

Выполнение первого требования обеспечивает:

- адекватность базы данных технологическому процессу,
- связь пользователя с базой данных - это уменьшает объем вводимой им информации, сокращает число ошибок и дает обзор совершенной работы.

Выполнение второго требования обеспечивает гибкое управление производством или обслуживанием, так как система позволяет получить обзор текущего состояния ресурсов и продуктов.

Выполнение третьего требования позволяет повысить качество принимаемых решений, выявить тенденции развития производства и обеспечить отчетность.

Выполнение требования эргономики является необходимым условием выполнения первых трех.

Из четырех требований, описанных выше, вытекают четыре типа ввода-вывода:

I) произвольный запрос

2) технологический макет

3) технологический запрос

4) технологический отчет

Технологический макет рассматривается в [3].

4. Произвольный запрос

Любые произвольные запросы к базе формулируются на языке запросов. Формально каждый запрос описывается следующим образом:

запрос ::= последовательность команд

последовательность команд ::= команда команда ...

команда::= открытие файлов / слияние файлов / / сортировка данных / выборка данных / арифметические действия над данными / вывод данных

открытие файлов::= ключевое слово для открытия, последовательность имен файлов

ключевое слово для открытия::= FILE

последовательность имен файлов::= имя файла имя файла ... имя файла::= слово в языке СКОРОПИСЬ

слияние файлов::= ключевое слово для слияния, повторяющее определение

повторяющее определение::= имя файла, ключевые слова отметки имя поля, имя общего поля

ключевое слово для слияния::= MERGE

имя файла::= слово в языке СКОРОПИСЬ

ключевые слова отметки имя поля::= BY KEY

имя поля::= слово в языке СКОРОПИСЬ

сортировка данных::=ключевые слова для сортировки,имена ключей

ключевые слова для сортировки::= SORTED BY

имена ключей::= имя ключа

имя ключа::=слово в языке СКОРОПИСЬ

выборка данных::= ключевое слово для выборки, термы

ключевое слово для выборки::= FIND

термы::= терм, терм ...

терм::= условие, знак логической операции

условие::= имя поля, знак условия, значение

знак логической операции::= & /V
имя поля::= слово в языке СКОРОПИСЬ
знак условия::= / = /< >/> / < /> =/<=/
значение::= число / слово в языке СКОРОПИСЬ

арифметические действия над данными::= ключевое слово (обозначающее вычисляемое поле), арифметическое выражение
ключевое слово (обозначающее вычисляемое поле)::= BREAK

арифметическое выражение::= имя поля-I, операция, имя поля 2, знак равенства, имя поля 3
имя поля::= слово в языке СКОРОПИСЬ
операция::= / + / - / * / : /
знак равенства::= ! =
выдача данных::= ключевое слово для отметки вывода / / имя поля / все поля /
ключевое слово для вывода::= DISPLAY
имя поля::= слово в языке СКОРОПИСЬ
все поля::= ALL

5. Технологические запрос и отчет

Результатом технологического запроса и отчета является получение данных из базы данных в требуемом виде.

5.I. Техника создания технологических запросов и отчетов

В системе разработаны средства создания запросов. В основу разработки этих средств были положены 2 принципа:

I) изменяемость макетов;
2) однозначное определение запроса к базе по описанию макета.

Средства создания запросов реализованы в виде подсистемы, состоящей из следующих компонентов:

- создание макета
- редактирование макета
- удаление макетов
- описание полей (окон) макета
- редактирование описания.

Создание макета. Процесс создания макета состоит из четырех шагов.

Первый шаг: Макету присваивается имя, которое является словом в языке СКОРОПИСЬ длиной I - 3I символов.

Второй шаг: На экране терминала рисуется макет документа. Макет документа состоит из постоянных текстов и окон. Окном называется область макета, через которую производится ввод-вывод данных. Поля, отводимые под окна, обозначаются специальными символами, а именно:

а) окна, входящие в заголовок, обозначаются символом ' ;

б) Q - обозначает окна, входящие в повторяющиеся строки;

в) @ - обозначает окна, входящие в концовку;

В системе предусмотрены средства для введения широких макетов (то есть, не помещающихся на экран в ширину). При нажатии кнопки 'RECALL' макет сдвигается на 52 позиции влево, то есть на экране появляется "заэкранная" область. Нажатие кнопки 'EDIT' возвращает экран обратно. Второй шаг завершается нажатием кнопки 'RUN' или 'PRINT'. При нажатии кнопки 'PRINT' шаг завершается вместе с выводом макета на печать.

Третий шаг: Структирование нарисованного макета. На этом шаге система задает последовательность вопросов, касающихся позиций начала и конца структурных частей макета. Обозначение начал и концов производится установкой курсора на начало или конец описываемой части. Имеются возможности для исправления разбиения на части (клавиша 'HALT/STEP').

Четвертый шаг: Сохранение макета и автоматический анализ. Шаг инициализируется нажатием клавиши 'SAVE'. При этом макет анализируется, определяются позиции и тип окон на экране, осуществляется контроль непротиворечивости макета и запись макета на внешний носитель, если макет непротиворечив.

Редактирование макета. Созданный макет можно отредактировать, выбрав режим "редактирование макета". После этого система задаст вопрос: Имя макета? на который следует ответить именем макета, требующего исправления. Шаги редактирования макета совпадают с шагами создания макета.

Удаление макета. Выбор режима "удаление макета" позволяет удалить ранее созданный макет по имени.

Описание окон макета. К описанию окон приводит выбор режима "описание полей". При этом на экране возникает макет, где окна отмечены белым фоном. Первый пробел описываемого окна мигает.

На нижней, зарезервированной строке экрана под макетом пользователь формулирует вопрос на языке запросов. Этот запрос определяет данные, помещаемые в окно. Каждая команда завершается нажатием кнопки 'RUN'.

Редактирование описания. Выбор режима "редактирование описания" разрешает изменять описание окон макета. После этого система задает вопрос: Имя макета? на который следует ответить именем макета, требующего исправления в описании данных.

Техника редактирования аналогична технике описания данных. Если нет необходимости в изменении определения данных, нажать кнопку 'RUN'.

Литература

1. Мартин Дж. Программирование для вычислительных систем реального времени. - М.: Наука, 1975.-С. 9.

2. Мартин Дж. Планирование развития автоматизированных систем. Финансы и статистика. - М., 1984.-С. 131-140.

3. Лумберг Т.А., Распель П.У. О разработке технологических структур для систем реального времени. (См. наст. сб. с. 100-113)

4. Блэкман М. Проектирование систем реального времени. - М.: Мир, 1977. - С. 29.

5. Ramamritham. Correctness of a distributed transaction systems// Inform. Systems. - Vol. 8, N4. - P. 309-324.

L. Elmik, M. Roost

About Real-Time Data Base

Abstract

In this paper some of the properties of real-time data bases and real-time information systems' input-output special features are considered.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК.681.3

Т.А. Лумберг, П.У. Распель

О РАЗРАБОТКЕ ТЕХНОЛОГИЧЕСКИХ СТРУКТУР ДЛЯ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Введение

Персональные компьютеры служат технической базой для вынесения ресурсов ЭВМ и современных методов обработки данных далеко за пределы больших вычислительных центров. При этом возникают новые требования в сфере эксплуатации ЭВМ. От качества созданного программного обеспечения зависит, станет ли персональная ЭВМ информационным ядром гибких систем автоматизации в народном хозяйстве [I].

В статье рассматриваются вопросы разработки технологических структур для систем реального времени, то есть проблемы управления технологиями и макетного обмена, который является расширением технологической структуры.

I. Управляющая система интерактивной инфосистемы

Большинство инфосистем состоит из ряда компонентов, которые осуществляют стандартные операции. Зачастую разница между двумя инфосистемами состоит лишь в последовательности выполнения обрабатывающих операций. В некоторых случаях, когда два предприятия или даже отделения одного предприятия обрабатывают информацию в основном с одинаковой структурой, одна и та же инфосистема тем не менее не подходит в силу различий инфраструктуры предприятий.

Если мы имеем дело с системой пакетной обработки, то в некоторых случаях подобную конфликтную ситуацию можно ликвидировать с помощью видоизменения работы оператора ЭВМ. Но при обработке информации интерактивной системой все от-

ношения между подсистемами, все структуры и их последовательности должны быть определены системными средствами. Кроме того, должна быть обеспечена защита как информации, так и технологических операций от "случайного пользователя". Таким образом, чтобы обеспечить совпадение структуры реальной объективной системы и инфосистемы, нужно создать средства, которые делают это возможным - определить структуру соответствующей инфосистемы, исходя из свойств реальной объективной системы.

В большинстве случаев такие управляющие средства реализованы фиксированными программами, которые подходят только для конкретной объективной системы. Это создает трудности внедрения одной и той же инфосистемы в разных условиях, так как управляющие средства такого типа очень жесткие и при каждой конкретной реализации требуют перепрограммирования довольно большого объема.

Чтобы избежать этого, нужно создать управляющую систему, которая дала бы возможность настраивать инфосистему на конкретную объектную систему с минимальными затратами труда. Параметрами настройки такой системы является формализированное описание структуры конкретной объектной системы - описание технологии обработки информации.

2. Сценарий технологии

Каждая технологическая операция на любом предприятии состоит из некоторых работ, которые надо выполнить. Таких операций довольно много. Фиксирование операций поставляет и информационные данные, регистрация и обработка которых создает информационное отображение технологического процесса: заполнение документов, сохранение документов, сводка документов, составление отчетов.

Если эти процедуры проделать с помощью вычислительной машины, то осуществляется это запуском нужных программ в нужном порядке. Тем самым достигается системное соответствие реальной и программной технологий. Программная технология должна быть отображением реальной.

Описание технологии обработки информации называется СЦЕНАРИЕМ технологии. Сценарий описывает соотношения под-

систем и прикладных программ, из которых состоит интерактивная инфосистема. При этом порядок выполнения подсистем не должен быть всегда одинаковым, его можно выбирать в ходе работы из фиксированных вариантов. Такая выборка должна производиться как системно, так и вручную - в ходе диалога системы и пользователя.

Системным выбором реализуются те связи системы, которые не подлежат произвольному выбору. Эти связи всегда одинаковы и зависят только от результатов выполнения подсистемы. Например, если выполнение какой-нибудь подсистемы кончится ошибкой, то она должна выдать соответствующий признак, а управляющая система должна проанализировать признак и выбрать нужное направление работы - обработка ошибки.

Частным случаем системного выбора направления работы является выбор "по умолчанию", другими словами, имеется только одна возможность продолжения работы, которая и выбирается.

Ручным выбором реализуются связи системы, которые неоднозначны, т.е. допускают произвольный выбор варианта работы. Ручной выбор представляет собой диалог между человеком и ЭВМ.

Если диалог инициируется ЭВМ, то машина сама "говорит" оператору, что делать. Такой диалог менее гибок потому,что оператор должен следовать предписанной последовательности действий. Однако при отображении реальной технологии это является даже преимуществом, так как дает возможность автоматически соблюдать технологию. Таким образом, диалог человека и ЭВМ целесообразно организовывать в виде выбора с меню и в виде вопроса-ответа. Используя такой диалог, можно обеспечить полное выполнение сложных работ. Резко уменьшается вероятность ошибок [2].

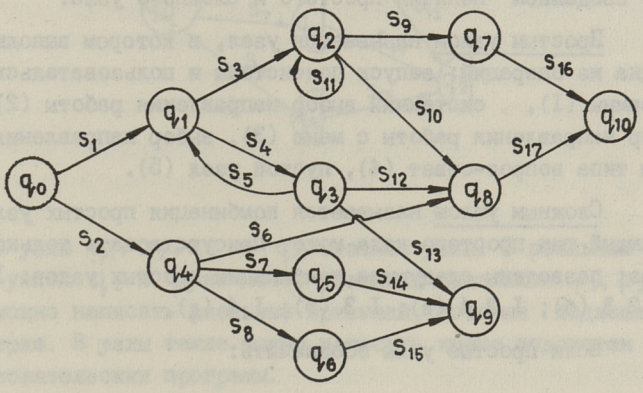
Работа над сценарием может начинаться после того, как инфосистема спроектирована и разделен на функциональные подсистемы.

В первую очередь объектную систему нужно разделить на технологические операции. На каждую технологическую операцию составляют отдельный сценарий. Для этого технологию разбивают на отдельные технологические узлы. В каждом узле

можно запустить либо какую-нибудь подсистему или пользовательскую программу, либо сделать выбор направления работы в другие узлы. В то же время в данном узле могут быть проделаны обе операции.

Таким образом, сценарий технологий представляет собой ориентированный граф. Узлами графа являются выполняемые функции (подсистемы или пользовательские программы) и/или выборы направления работы. Какая-либо конкретная технологическая операция представляет собой путь на графе сценария. Частичное перекрытие разных технологических операций неизбежно. В пределах одного сценария может существовать несколько сходных узлов [3].

Посмотрим пример графа сценария:



Здесь узел q_0 - начальный узел сценария и узлы q_9 и q_{10} - конечные узлы. Предположим, что в узле q_0 не выполняется ни одной подсистемы. В таком случае направления s_1 и s_2 могут быть только ручным выбором типа вопрос-ответ, так как вариантов выбора только два. Предположим также, что в узлах q_3, q_4, q_5, q_7, q_6, q_8, q_9 подсистемы выполняются. В подсистемах, выполняемых в узлах q_2 и q_3 вырабатываются признаки. В узле q_2 существует возможность, что признак не вырабатывается. В таком случае все направления работы s_5, s_{12} и s_{13} выбирают по выработанным признакам, но из направлений s_9, s_{10} и s_{11} одно должно быть направлением "по

умолчанию" - для случая, когда признак не вырабатывают. Выбор направления работы в узле q_4 ручной - с меню, так как вариантов выбора более двух и в узле не выполняется ни одна подсистема. Направления с узлов q_5, q_6, q_7 и q_8 - направления "по умолчанию", так как в каждом случае существует только одна возможность направления работы.

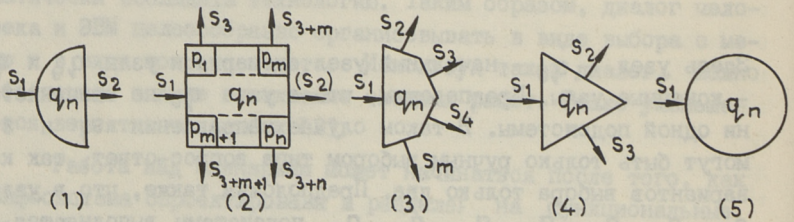
Как уже сказано, узлы q_9 и q_{10} конечные. Конечными узлами технологии называют узлы, из которых не выходит ни одного направления работы. После выполнения операции в конечном узле управление передается опять начальному узлу.

Отсюда следует, что в графе имеются узлы с неодинаковыми структурами. Их одинаковое обозначение делает граф технологии неоднозначным. Чтобы внести ясность, разные типы узлов нужно пометить различно. Проще всего добиться этого введением понятия простого и сложного узла:

Простым узлом называется узел, в котором выполняется одна из операций: запуск подсистемы и пользовательской программы (I), системный выбор направления работы (2), выбор направления работы с меню (3), выбор направления работы типа вопрос-ответ (4), пустой узел (5).

Сложным узлом называется комбинация простых узлов, где каждый тип простого узла может присутствовать только один раз; позволены следующие комбинации простых узлов: I,2 (а); I,2,3 (б); I,2,4 (в); I,3 (г); I,4 (д).

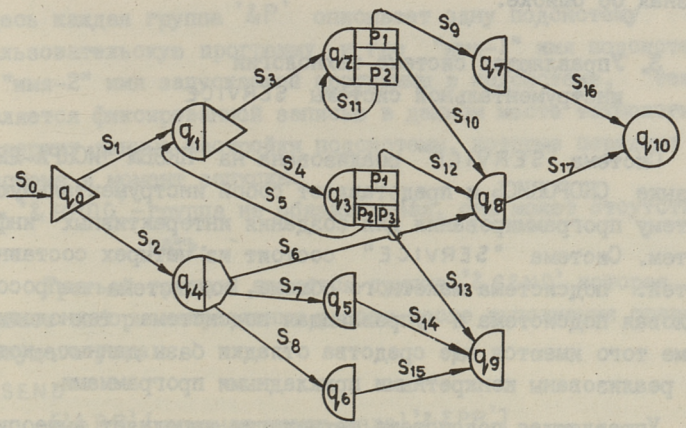
Если простые узлы обозначить:



тогда дуга S_1 - всегда вход в узел. В вариантах (I) и (2) дуга S_2 является направлением "по умолчанию" и может отсутствовать в варианте (2), когда в подсистеме, которая

выполнялась прежде выбора (2), в любом случае вырабатывается признак выбора направления работы. Пустой узел (5) всегда может быть только конечным узлом.

Если теперь нарисовать граф, предъявленный выше, то он выглядит так:



Здесь узлы q_1, q_2, q_3 и q_4 сложные узлы. В реальном графе в узлах q_2 и q_3 на место условных признаков p_1, p_2 и p_3 можно написать реальные признаки, которые подлежат проверке. В узлы также можно написать имена подсистем или пользовательских программ.

Для того, чтобы так просто и динамично определять структуру подсистем инфосистемы, все подсистемы должны удовлетворять некоторым условиям:

а) все подсистемы (и пользовательские программы) должны представлять собой временно и технологически независимые единицы, т.е. каждая подсистема должна быть логическим целым, которое нельзя разбить на части без изменения ее назначения;

б) все подсистемы должны иметь буфер для приема данных от управляющей системы, а также для передачи данных ей;

в) каждая подсистема, которая в ходе работы создает глобальную область данных, перед окончанием работы должна уничтожить её;

г) ни одна подсистема не должна заканчивать работу аварийной ситуацией, а все ошибочные ситуации должны анализироваться в самой подсистеме;

д) если система заканчивает работу после анализа аварийной ситуации, то она должна дать управляющей системе признак об ошибке.

3. Управляющая система технологии инструментальной системы "SERVICE"

Система "SERVICE" реализована на ПКЭВМ "ИСКРА-226" в языке СКОРОПИСЬ и представляет собой инструментальную систему программирования для создания интерактивных инфосистем. Система "SERVICE" состоит из четырех составных частей: подсистема макетного обмена, подсистема запросов, файловая подсистема и управляющая подсистема технологии. Кроме того имеются еще средства отладки базы данных, которые реализованы конкретными прикладными программами.

Управляющая подсистема технологии выполняет вышеописанную функцию, т.е. позволяет все подсистемы и прикладные программы соединить в целостную инфосистему. Подсистема реализована настраиваемой системой, которая имеет свой входной язык, который допускает описание любой реальной технологии. Ниже описываются принципы работы подсистемы и её входной язык. При изложении синтаксиса входного языка использован метаязык, являющийся расширением нотаций Бэкуса.

Входной язык управляющей подсистемы технологии построен по узловому принципу, т.е. описание каждого узла создается и сохраняется на диске отдельно. Входной язык описания технологии дает возможность создать описания как простых, так и сложных узлов.

Описание каждого узла начинается группой '&HEAD'. Синтаксис группы '&HEAD' следующий:

'&HEAD' текст '&ENDH'.

Текст между операторами '&HEAD' и '&ENDH' должен коротко характеризовать операции в узле. Его используют при выводе каталога узлов технологии при отладке, а также для составления меню в ходе работы.

'&HEAD' группа обязательная.

Второй группой является группа '&PROG', которая описывает выполняемые в данном узле подсистемы и пользовательские программы:

'&PROG'{'&P' имя-I имя-2 текст'&EP'}'ENDP'.

Здесь каждая группа '&P' описывает одну подсистему или пользовательскую программу, и где "имя-I" имя подсистемы и "имя-2" имя запускаемой программы в подсистеме, "текст" является фиксированной записью в данном месте технологии и содержит данные настройки подсистемы, которые передают подсистеме в момент запуска.

'&PROG'-группа не обязательная, она может отсутствовать.

Третьей группой является группа '&SEND', которая описывает выборы направления работы после выполнения операции текущего узла:

```
'&SEND'
    ['&PR'{признак номер-узла}'&EPR']
    ('&H'{nr}[E|F|P]'&EH'!
    '&Q'текст '&QQ' ответ-I номер-узла-I
     ответ-2 номер-узла-2 '&EQ'!
     '&S' номер-узла '&ES')
'&ENDS'.
```

Подгруппа '&PR' группа системного выбора направления работы. За оператором '&PR' следуют пары "признак номер-узла" - в случае, если предыдущая подсистема выработала признак, его ищут в списке между операторами '&PR' и '&EPR' и передают управдение узлу, номер которого стоит за признаком. В случае, если признак в списке не находится, выбирают направление работы по следующей подгруппе направления ('&H','&Q' или '&S'), если оно существует. В противном случае дается управление первому узлу технологии.

Подгруппа '&H' описывает выбор с меню. За оператором пишутся все номера узлов, которые нужно выбирать из меню. В ходе работы на дисплее составляют меню с текстов '&HEAD'-группы тех узлов, которые перечислены в подгруппе

'&H'. Добавление в список номеров узлов символа 'E' добавляет в меню пункт "Выход с системы", символа 'P' - "Выход с меню", то есть возвращение к предыдущему меню, и символа 'F' - "Возвращение к началу технологии".

Подгруппа '&Q' осуществляет выбор типа вопрос-ответ, где "текст" между операторами '&Q' и '&QQ' представляет собой вопрос. Между операторами '&QQ' и '&EQ' находятся две пары "ответ номер-узла", т.е. разрешенные ответы и номера узлов, куда передадут управление после ответа на вопрос.

Последняя подгруппа направления работы группа '&S' , которая осуществляет направление "по умолчанию". Когда направление работы выбирается на основе группы '&S', то управление передают узлу, номер которого непосредственно следует за оператором '&S'.

Группа '&SEND' не обязательна, она может и отсутствовать.

Последней группой описания узла является группа'&TEXT'

'&TEXT'[признак текст]'&ENDT'.

Группа дает возможность после выполнения в узле описанных операций, прежде чем управление передадут другому узлу, вывести на экран текст, написанный между операторами '&TEXT' и '&ENDT' "признак" дает возможность остановки системы на некоторое время:

а) если признак цифра от I до 65 000, то производится остановка работы на секунды, равные признаку;

б) если признак символ '*' (звездочка), то после вывода определенного текста выводится еще текст "для продолжения работы нажмите на любую клавишу!" и ожидается нажим клавиши. Когда признак отсутствует, остановка не производится.

Группа '&TEXT' не обязательна.

В интересах гибкости системы особый статус дан нолевому узлу. Узел с номером "ноль" всегда должен существовать в технологии. В нем могут существовать только группы '&HEAD', текст которой обозначает имя системы, и подгруп-

па '&S' группы '&SEND', которая показывает на действительный начальный узел технологии. Это дает возможность преодолеть трудности поиска первого узла технологии при запуске системы, а также при возвращении с конечного узла в начало технологии. Кроме того очень удобно переопределять первый узел технологии.

4. Организация обработки документов - назначение, требования и свойства

Средства обработки документов предназначены для организации ввода-вывода данных на видеотерминал.

Если иметь в виду конечного пользователя, то его работа с документами на ЭВМ должна быть максимально приближена к привычным действиям без ЭВМ.

Техника ввода-вывода данных на дисплей в форме документов способствует передачи данных в больших объемах, при этом значительно уменьшается вероятность ошибок. Ведь обработка документов с помощью ЭВМ на рабочем месте конечного пользователя - это по существу диалог, проводимый необученным оператором.

Первое, что требуется от необученного оператора, это избегать использования предложений, которые ЭВМ не сможет интерпретировать. Наиболее эффективным путем для этого является ограничение ввода. Одним способом ограничения ввода является применение техники "выбора меню", то есть частный случай диалога, инициируемого ЭВМ. Другой способ состоит в том, чтобы заставить оператора "заполнять форму" на экране дисплея. Изображение дается с пропусками, которые оператор должен заполнить [2].

В реальной жизни работа с документами связана с некоторым технологическим процессом, который всегда носит конкретный характер. Следовательно, системные программные средства должны быть настраиваемыми и в то же время эффективными. Они должны позволять обрабатывать разные документы в разных условиях, либо не требуя при этом никаких изменений в самой системе, либо с минимальными и легко осуществимыми изменениями.

Учитывая приведенные выше рассуждения, можно выделить особенности средств обработки документов. Главными отличительными чертами являются:

I) широкий круг конечных пользователей;

2) строгие требования к форме представления информации (документами), которые меняются во времени;

3) необходимость проверки большого количества входной информации.

Средства обработки документов должны удовлетворять следующим требованиям:

I) гибкость и комплексность взаимосвязи с пользователем, которая должна обеспечить удобное и естественное обращение к системе со стороны любого класса пользователей;

2) эффективность доступа к данным; см. [4]

3) возможность проверки и модификации вводимой информации;

4) настраиваемость и модифицируемость.

Средства обработки документов можно рассматривать как самостоятельные средства. Но так как обработка документов является частью технологического процесса в масштабе реального времени, то эти средства целесообразнее рассматривать подчиненными управляющей системе технологии. Практически это значит, что документ, который следует заполнять в технологическом процессе, надо связывать с конкретным технологическим узлом в структурном описании этого процесса. Связь устанавливается по имени документа в описании технологического узла.

В этом контексте мы говорим уже не о документе в обычном смысле, а о так называемом технологическом макете. Под технологическим макетом подразумевается средство перевода реальных технологических операций на язык системы.

Технологический макет должен содержать нужную информацию для выполнения технологической операции и являться носителем информации, посредством которого передаются результаты технологической операции в базу данных.

Первую функцию выполняет, например, изображение документа, который содержит и выведенную информацию в поля вывода данных.

Вторую функцию выполняют поля ввода данных, так как содержимое заполненных полей ввода данных - это по существу результаты технологической операции.

Характер, объем и специфика полей данных (например связи с базой данных) в технологическом макете определены технологической операцией.

Во время работы с технологическим макетом происходит интерактивный обмен данных между базой данных и пользователем посредством видеотерминала.

Таким образом средства обработки документов являются в инфологическом контексте средствами макетного обмена.

5. Средства макетного обмена

Особенностью средств макетного обмена является использование преимуществ видеотерминала для наглядного представления информации. Это достигается путем вывода на экран дисплея изображения, соответствующего обрабатываемому документу, на котором видны названия полей данных, поясняющие смысл вводимой информации, и поля данных, отмеченные специальными символами. При этом, в отличие от традиционных средств обмена, где единицей обмена является строка символов, возможно использовать весь экран дисплея для представления документа или страницы документа. Все это позволяет при помощи вышеупомянутых средств обрабатывать документы с практически произвольной формой. Автоматически проверяется правильность ввода данных.

Средства макетного обмена состоят из набора программ, которые образуют по их функции две основные части.

Макетный редактор - это средство для создания и описания макета, а также его исправления. Макет соответствует одному документу, количество страниц в макете неограничено. Под страницей макета понимается образ экрана, который содержит изображение документа (или его части) и поля данных. Последние отмечены специальными символами: Q (текст), ↑

(целое число), @ (вещ. число). Количество специальных символов в последовательности этих символов определяет длину этого поля данных.

Это значит, что одновременно с созданием изображения документа на экране описывается система, местонахождение и длина полей данных, а также тип данных. При этом не нужен особый язык описания макетов. Дополнительно надо описывать только связи между полями данных и элементов в базе данных, то есть определить место хранения для каждой единицы вводимой или выводимой информации. Эти связи устанавливаются посредством спецдиалога.

Существуют еще служебные символы - знак % (конец заголовка макета), и прямые скобки [ ] (часть макета между ними - это повторяющаяся часть при вводе данных). При вводе данных повторяющаяся часть двигается вверх "под" заголовком, когда начинается новый цикл повторения и экран уже заполнен данными. Число повторений определяется прямо при вводе данных: ввод данных в пределах такой части макета осуществляется циклически до нажатия спецклавиши.

Макетный редактор производит анализ изображения документа на экране с отмеченными полями данных и составляет описание полей данных макета. Описание макета хранится на диске как единица хранения, имя которой соответствует имени макета.

Для исправления макета выводится на экран текущая страница макета, которую можно произвольно изменить - редактор создает новое описание этой страницы, которое записывается вместо старого.

Макетный драйвер - это средство активируется при вызове из управляющей системы технологий, и на экран выводится макет (первая страница) для ввода данных. Текущее поле данных отмечено белым фоном или выводится туда значение (поле вывода). Позиции экрана, занятые текстом документа - заблокированы: курсор перескакивает через них. Если в поле будут введены неправильные данные, на нижней строке экрана появится сообщение об ошибке, и курсор поместится на начало того же поля для повторного ввода. После ввода последнего значения в макет сеанс завершается. Введенные данные

передаются в базу данных соответственно связям и действие переходит в узел технологии, при выполнении которого макет был вызван по имени макета.

Литература

I. Громов Г.Р. Национальные информационные ресурсы: проблемы промышленной эксплуатации. - М.: Наука, 1984. - С. 7-8.

2. Мартин Дж. Системный анализ передачи данных - М.: Мир, 1975. - С. 10-95.

3. Петерсон Дж. Теория сетей Петри и моделирование систем. - М.: Мир, 1984. - С. 17-22.

4. Эльмик Л.Н., Роост М.Х. База данных для систем реального времени. (См. наст. сб. с. 90-98.)

T. Lumberg, P. Raspel

About Creating the Technological Structures for Real-Time Systems

Abstract

Organization of real-time data processing is a necessary pre-condition for automatization of current information processes.

This paper deals with some problems of creating the real-time data processing technology. For creating such a theory, requirements, to which the real-time data processing technology must correspond, are taken into consideration. At last the above-mentioned technology is described.

№ 645
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.06

Е. Бернштейн

СИСТЕМА МАКЕТНОГО ОБМЕНА МАО

Широкое распространение дисплеев позволило существенно упростить взаимодействие с ЭВМ и расширить круг ее пользователей. Одним из преимуществ дисплея является возможность наглядного представления вводимой и выводимой информации в виде, привычном для непрофессионального пользователя. Таким образом, следует реализовать эти возможности, разработав соответствующее программное обеспечение. В данной статье описывается система макетного обмена МАО, предназначенная для работы с документами или формами на экране дисплея. Идея состоит в том, что на экран выводится макет привычного пользователю документа. Макет состоит их поясняющих текстов (названий полей) и самих полей (окон), в которые помещается информация при обмене (при вводе пользователем, при выводе – прикладной программой). Таким образом, в отличие от традиционных средств обмена, где единицей обмена является строка символов, в данном случае единицей обмена данными становится весь экран.

Система МАО разработана для мини-ЭВМ СМ-4 (СМ-1420, СМ-1600) и работает под управлением операционной системы ОС-РВ. Ввиду большого разнообразия выпускаемых отечественной промышленностью терминалов, в системе предусмотрена автоматическая настройка на терминал пользователя (если этот терминал находится в списке известных системе терминалов), а так же разработана процедура внесения нового терминала в список известных.

I. Общее описание и состав системы

Система МАО состоит из набора программ и подпрограмм, облегчающих прикладной программе интерактивный обмен с использованием дисплея, в систему входят:

I. Макетный редактор: программа для создания и описания макета, а также его корректировки. Макетный редактор предоставляет возможность проектировать и исправлять макеты документов непосредственно на экране, в экранном режиме, избегая тем самым, создания языка описания макетов. Описание макета хранится в файле на диске.

2. Макетный драйвер. Это набор подпрограмм, вызываемых из прикладной программы. При вызове подпрограмм макетного драйвера на экран выводится заранее подготовленный макет для работы с ним оператора.

2. Макетный редактор

Макетный редактор - это программа, позволяющая проектировать, описывать и исправлять макет в двух режимах: командном и экранном.

В командном режиме исполняются команды чтения макета с диска, записи макета на диск, очистки буфера, перевода текущей страницы, вывода макета на печать и другие.

В экранном режиме пользователь видит перед собой на экране страницу макета и с помощью указателя (курсора) и клавиатуры (включая функциональные клавиши), производит редактирование. После редактирования макета пользователю предоставляется возможность описать атрибуты полей, облегчая в дальнейшем макетному драйверу контроль ввода.

3. Макетный драйвер

Разработчик прикладной программы использует макетный драйвер для ввода записей в программу, обновления и вывода записей, вызывая подпрограммы макетного драйвера. Макетный драйвер позволяет:

I. Открывать макет, подготовленный редактором.

2. Вводить записи - при этом на экран терминала выводится макет, и ввод записей заключается в заполнении окон, отмеченных специальным образом.

3. Обновлять записи - на экран выводится макет, в котором окна заполнены значениями полей обновляемой записи. При вводе данных в окна значения полей обновляются.

4. Выводить записи – на экран выводится макет, в окна которого помещены значения полей выводимой записи. Окна при этом заблокированы от исправлений.

5. Очищать экран.

6. Использовать нижнюю строку экрана для вывода подсказок оператору.

7. Выводить документ на печатающее устройство.

4. Пример использования МАО

Так как МАО является инструментальной системой программирования, можно использовать в приложениях, требующих форматированного экрана. В качестве примера можно привести систему ПРОГЕН, разработанную на кафедре обработки информации ТПИ и внедренную в НТПО "МИСТРА".

Система ПРОГЕН предназначена для решения задач бухгалтерского и оперативного учета, а также для генерации других приложений. Система ПРОГЕН освобождает пользователя от программирования в традиционном смысле. Центр тяжести переносится на проектирование входных и выходных машинных документов. Программные средства системы ПРОГЕН можно разделить на три группы.

I. Программы загрузки и обновления базы данных.

2. Генерация отчетов,

3. Обслуживающие программы.

Система МАО используется на двух уровнях:

I. Программы ПРОГЕНА используют МАО для диалога с конечным пользователем. Пользователь задает управляющую информацию для описания входных данных, для описания выходных отчетов, для описания структуры базы данных и т.д., заполняя поля в заранее подготовленных макетах диалога. Например, для описания выходного отчета на экран выводится следующая картинка: (рис. I) в которой пользователю предлагается заполнить подчеркнутые поля.

2. Пользователь подготавливает макеты своих входных и выходных документов макетным редактором. В дальнейшем эти макеты используются программами загрузки базы и вывода отчетов в процессе использования базы данных.

Форматирование управляющей информации для генератора отчетов

Наименование отчета ....
Название выходной формы ....
Имя файла выводимых данных ....
Имя файла описания данных..... .....

Суммарные признаки

Число суммируемых признаков ....
Номера суммируемых признаков... ....
Число строк на странице ....

Рис. I.

E. Bernshtein

The Forms Input-Output System MAO

Abstract

There are numerous instances of commercial, scientific and industrial applications in which formatted video screen provides excellent benefits. In this paper the MAO system, which provides a screen management software for applications using a wide range of terminal types, is described. MAO consists of two parts: screen forms driver and forms editor. The new terminals customization procedure is provided by the system. The paper also gives an usage example.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

Т. Ваппер, Ю. Лааст-Лаас,
А. Урвак, Л. Эльмик

ПРАКТИЧЕСКИЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Ниже рассматриваются некоторые проблемы проектирования информационных систем, которые возникли в практике при выполнении договорных работ кафедры обработки информации ТПИ. В частности, затрагивается проблема нахождения терминологического взаимопонимания между проектировщиками и заказчиками: изучение информационных потребностей заказчиков и конкретизация проектной задачи; определение генеральной структуры системы, т.е. разбивка информационной системы в раздельно реализуемые подсистемы. Перечисленные и многие другие вопросы необходимо решать в исходной стадии проектирования, от их решения во многом зависит насколько удачен будет проект, а также его качество, стоимость и время выполнения.

Нами применяется смешанная методика системной работы. Под системной работой здесь понимается комплекс мероприятий по изучению существующей информационной системы и по диагностике ее недостатков, по анализу действительных информационных потребностей ее пользователей, а также по разработке логического проекта новой информационной системы. Основные методологические источники и компоненты нашего подхода следующие: инфологический подход (см. напр. [1, 2]), методика стратегического проектирования информационных систем BSP [3], настенная техника [4], система DAJE [5].

Одним из основных методических принципов исследования предприятий и организаций является выделение объект-системы и монитор-систем. Объект-система является наблюдаемой и/или управляемой системой, а монитор-системы являются системами наблюдения и контроля, соответствующие различным аспектам управления и исследования. Объект-система выполняет основ-

ные функции исследуемой организации и охватывает комплекс ресурсов, которые либо используются, либо создаются в соответствующих процессах.

Монитор-системы выполняют вспомогательные функции (бухгалтерский учет, составление статистических отчетов и т.д.). Основным ресурсом монитор-системы является информация. Монитор-системы являются как бы искусственным дополнением к объект-системе, их задачи меняются быстро, их структура и функционирование до определенной степени независимы от структуры и функционирования объект-системы.

РЕКОМЕНДУЕМ

начинать проектирование информационной системы не с исследования и определения задач обработки данных и управления, а с определения в первую очередь объект-системы, с изучения ее структуры и поведения и на основе этих результатов строить ее информационную модель.

В практике проектирования информационных систем необходима согласованная деятельность и стремления двух сторон: заказчика - будущего пользователя системы и проектировщика. Имеются разные методики, которые и по-разному решают вопрос участия будущего пользователя в процессе проектирования: от методик, предусматривающих включение пользователей в состав проектной группы [4] до таких, которые основываются только на обмене документации между сторонами. Уже во время первых встреч с заказчиками должна возникнуть "контактная область", которая становится основой их дальнейшего сотрудничества.

Архитекторы, как будто, находятся в более благоприятных условиях, чем проектировщики информационных систем. Их заказчики точно знают, для чего им нужен проектируемый объект, и, как правило, предъявляют ему определенные функциональные и технические требования. Проектировщик информационной системы в значительной мере должен угадывать потребности пользователя. О проблемах заказчиков он как бы вынужден знать больше, чем знают заказчики сами. Кроме того, он должен предвидеть тенденции изменения этих потребностей и учитывать это в проекте.

Ситуация не будет столь безнадежной и противоречивой, если для проектирования информационной системы выбран подход, основывающийся на моделировании объект-системы. Концептуальное описание объект-системы почти всегда представляется с использованием только простых и общепринятых понятий. На этой почве быстро находится общий язык между проектировщиками и пользователями, способствующий плодотворной совместной работе. На основе концептуального описания объект-системы наглядно вырисовываются и проблемы управления объект-системой, выявляются слабые стороны существующей информационной системы. В ходе интервью с разными специалистами организации заказчика проектировщик-"инфолог" концентрирует все свое внимание на объект-системе и (в начальной стадии проектирования) игнорирует почти всю информацию о вопросах обработки данных и управления, о содержании конкретных документов и о конкретных задачах пользователей. Эти сведения при этом являются лишь ориентирами для рационального определения объект-системы.

РЕКОМЕНДУЕМ

вести звукозапись разговоров с будущими пользователями системы. Это существенно сэкономит время совместной работы и повысит результативность бесед. Одни и те же фонограммы становятся ценными источниками сведений на разных стадиях проектирования, поскольку во время интервью высказываются предложения, замечания и мысли не только по узко заданным вопросам.

РЕКОМЕНДУЕМ

сначала выяснить то, о чем нужны сведения пользователю. Не следует слишком рано сосредотачивать внимание на том, какая именно информация ему нужна, как ее надо обрабатывать и в каком виде представлять.

На исходном этапе проектирования системы интересы проектировщика должны быть как можно больше связаны с объект-системой и как можно меньше зависеть от существующей систе-

мы обработки данных (монитор-системы). В пользу такого подхода говорят следующие аргументы: I) объект-система является общей для всевозможных аспектов ее наблюдения и управления; 2) архитектура и механизм поведения объект-системы более стабильны и поэтому она поддается описанию и моделированию легче, чем монитор-системы.

Моделирование объект-системы в теории, а также и в выбранных учебных примерах кажется очень естественным и простым, но в практике проектирования появляются существенные трудности. Объект-система понятие абстрактное, ее протяжение и построение в каждом конкретном случае выясняется только в процессе ее информационного моделирования. Провести физический осмотр целой объект-системы обычно невозможно. Обследование существующей информационной системы, изучение функций, задач и потребностей ее пользователей является неизбежным. Однако эти исследования должны иметь лишь вспомогательный характер. Их первичной целью является определение границ и структуры объект-системы, установление разумной степени детальности ее моделирования.

Новая информационная система никогда не строится на пустом месте. Существует старая система, которая явно не удовлетворяет требованиям (раз ее необходимо заменить), но она имеет глубокие корни как в человеческом сознании, так и в канцелярской бюрократии. В связи с этим обратим внимание на одно неприятное явление, которое, по нашим наблюдениям, снижает эффективность многих систем обработки данных, реализованных на ЭВМ. Такое явление сохранения (или точнее: механического копирования) в новой среде некоторых элементов старой информационной системы, потерявших свое первоначальное значение, называем "пересаживанием". Пересаженные элементы появляются как никому ненужные реликты и иногда причиняют значительный ущерб. Документы, например, иногда заменяются эквивалентными записями файлов или баз данных, ручные операции обработки бумаг заменяются соответствующими программами ЭВМ, не прибавляя к логике этого процесса ничего существенно нового. При этом значительные резервы возможностей автоматизированной обработки данных остаются не затронутыми, а в то же время имеющиеся (дефицитные) ресурсы ЭВМ часто используются нецелесообразно. Избыточность, сложность и негибкость таких систем очевидна.

Суть проблемы в том что при замене ручного человеческого труда работой ЭВМ не только можно, но (для получения качественного сдвига в информационной деятельности) и необходимо менять логику и постановку самой работы.

Пересаживание не всегда проявляется так явно. Обычно проводится проектирование системы с учетом традиционных требований машинной обработки данных, но явление пересаживания при этом не обязательно исчезает. Иногда некий набор задач (обычно называемый "функциональной подсистемой", но в действительности не представляющий собой ничего кроме задач, выделенных из старой информационной системы) рассматривается и реализуется как самостоятельный блок информационной системы. Позднее появляются сложные проблемы перестройки этих блоков и согласования их между собой. Переделка и усовершенствование одного блока влечет за собой изменения в других и т.д. Словом, автоматизированная информационная система вечно находится в стадии перестройки и ремонта.

Что, собственно, неправильно? - Логическая ошибка была допущена уже на самой начальной стадии проектирования, когда физические компоненты системы были определены соответственно произвольному выбору задач. Как видно, эти задачи время от времени меняются, они имеют сложные и динамичные связи между собой. Следовательно, и архитектура информационной системы соответственно становится сложной и динамичной, часто требующей перестройки.

Обратим внимание, что перестройка принципиально отличается от настройки. Перестройка каждый раз требует (частичного) разрушения системы, за которым следует этап ее восстановления, отладки, тестирования и внедрения. Настройка же осуществляется в процессе работы системы.

РЕКОМЕНДУЕМ

при моделировании объект-системы строго различать инфологический и даталогический уровни [2]. Перемеживание инфологических (концептуальных, семантических) проблем с даталогическими вопросами (вопросы языка представления (кодирования) информации и вопросы структур (макро- и микро-уровней), связанных с физической (носитель) и логической средой

(СУБД, напр.) представления информации), по нашему мнению, является главной причиной пересаживания.

Для декомпозиции информационной системы предлагаем принцип распределения информационной модели объект-системы в целостные подмодели. При этом имеется две возможности: "логическая" и "физическая" декомпозиция.

Физическая декомпозиция информационной базы и всей информационной системы соответствует определенному физическому делению объект-системы (территориальному, например). Информационная система объединения, например, может быть разбита в локальные системы по филиалам. Однако надо обратить внимание на то, что такая физическая декомпозиция, хотя и уменьшает физический объем данных, хранимых и обрабатываемых, в то же время не упрощает ни структуру информационной базы, ни функциональную сложность подсистем в сравнении с единой централизованной системой. Но для сохранения общего представления об объект-системе такая декомпозиция предполагает либо создание общей базы данных (дополнительно к распределенным базам), либо построение специального интегрирующего функционального модуля.

Логическая декомпозиция означает деление логической схемы информационной модели на относительно целостные и независимые подсхемы. Каждая подсхема имеет значительно более простую логическую структуру в сравнении с целой схемой. Определенные таким образом информационные подсистемы намного проще поддаются реализации. Особенной чертой каждой из них является ориентация на определенный, более узкий спектр приложений. В целях обеспечения интеграции данных для общих приложений, подчас, требуется частичное перекрытие схем различных подсистем. А это означает некоторую избыточность данных, проблем обновления и т.д. Но в данном случае ситуация, можно сказать, стабильная, т.е. она не выйдет из под контроля в процессе усовершенствования разных подсистем.

Логический и физический способы декомпозиции могут использоваться комбинированно. С точки зрения простоты реализации отдельных подсистем (имеется в виду простота струк-

тур данных и программного обеспечения) следует предпочитать логическую декомпозицию. Стремление к рассредоточению обработки данных, с другой стороны, лучше согласуется с физической декомпозицией.

Применяемую нами смешанную методику можно сравнить с решением огромного кроссворда. Эта методика характеризуется тем, что процесс системной работы не следует точному сценарию, а является до определенной степени "случайным", зависимым от порядка поступления сведений. В то же время, правильность отдельных решений и их совместимость с описанными раньше постоянно обеспечивается.

Отличной "контактной областью" между заказчиком и проектировщиком при этом является инфологическая (концептуальная) модель объект-системы. Эта модель постоянно усовершенствуется и уточняется в процессе системной работы. Уже в самом исходном интуитивном варианте она поддерживает взаимопонимание сторон.

Удобной и дешевой материальной основой для динамичного и наглядного представления промежуточных результатов, а также проектных решений системной работы, является "стенная" техника. Она обеспечивает коллективное взаимопонимание и информированность, и в то же время дает хорошую возможность наблюдать и управлять процессом проектирования, а также экономию труда и простоту внесения изменений.

Если проектирование информационной базы осуществляется методом инфологического моделирования объект-системы, а определение информационных подсистем проводится разбиением этой модели, то отпадают проблемы интеграции различных задач, приложений, частей информационной базы.

Литература

1. Laast - Laas J. Data Base infological design in practice // Transactions of Tallinn Technical University. - 1986. - N 614. - P. 107-112.

2. Лааст-Лаас Ю.Г. Проблемы инфологического проектирования структур данных для комплекса экономических задач // Тр. Таллинск. политехн. ин-та. - 1981. - № 511. - С. 3-13.

3. Мартин Дж. Планирование развития автоматизированных систем. //Финансы и статистика. -М., 1984.

4. Saaren - Seppälä, K. Seinätekniikka Systeemin suunnittelun opas. - Helsinki: Tietojenkäsittelyliitto, 1983.

5. Kaje, M. Nevalainen, R. DAJE-järjestelmä yrityksen johtamiseen ja kehittämiseen. Oy Dava-Instituutti Ab, Otakustantamo, 1986.

L. Elmik, J.Laast-Laas, A. Urvak,
T. Vapper

Practical Aspects of Information System Design

Abstract

A mixed strategy of information system design is used in the Department of Information Processing of the Tallinn Technical University. Some general standpoints of the mixed strategy are presented. Several practical advices for system engineering are given. It concerns, in particular, object-system determination, its modelling in data base and data base decomposition.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.016

А.В. Рензер

ФУНКЦИОНАЛЬНЫЕ АСПЕКТЫ АВТОМАТИЗАЦИИ ЭКОНОМИЧЕСКОГО АНАЛИЗА

Введение

Внедрение современных технологий в народном хозяйстве требует коренного изменения традиционной организации управления, в особенности перехода на автоматизированные формы управления безбумажным документооборотом и автоматизированными рабочими местами (АРМ) для всех видов конторских работ. Проведение экономического анализа является одной из основных функций управления, но вопросы его автоматизации изучены, пожалуй, меньше всего.

В статье рассматриваются проблемы проектирования автоматизированных инфосистем экономического анализа и предлагается общая функциональная модель разработок, реализующих функцию анализа на различных АРМ.

Методика автоматизации экономического анализа

Конечной целью автоматизации является повышение производительности труда персонала, занятого аналитическими работами. По [I, с. 30] имеется четыре способа обработки аналитической информации:

- ручной, который основывается на традиционном ручном труде работников аппарата управления;
- механизированный, где используются различные технические средства;
- автоматизированный, при котором ЭВМ принимает на себя наиболее трудоемкие и сложные процессы обработки и тем самым освобождает работников аппарата управления от однообразной и утомительной работы;

– автоматический, чисто машинный способ, при котором весь процесс обработки информации и воздействие на управляемый объект выполняются техническими средствами без вмешательства человека.

Последний способ обработки успешно используется при управлении технологическими процессами, для проведения же экономического анализа в традиционном смысле он еще не подходит. Первые два способа – ручной и механизированный – не представляются перспективными и поэтому в статье рассматриваются проблемы только автоматизированного экономического анализа.

Автоматизация должна начинаться с создания АРМ, которые являются компонентами автоматизированной системы и обеспечивают информацией работников аппарата управления. Отдельные АРМ соединяются физически и информационно, и составляют функционально единую систему. На уровне этой системы данные собираются, первично обрабатываются и хранятся в т.н. локальной базе данных. В дальнейшем такую систему будем называть системой нижнего уровня.

Система высшего уровня соединяет между собой несколько систем нижнего уровня, которые для нее являются функциональными подсистемами. Логично было бы создать для предприятия информационную систему, которая соединяет функциональные подсистемы кадрового и бухгалтерского учета, экономического анализа, планирования и т.д. При этом в общую информационную базу системы высшего уровня не попадает вся первичная информация, туда передаются только те сводные данные, которые нужны вне среды работы системы нижнего уровня.

Информационная совместимость систем нижнего уровня обеспечивается их проектированием по единому плану, по т.н. стратегическому плану данных. Этот план является развитием на уровне структур данных инфологической модели процесса производства. Отметим, что истинной основой всего процесса проектирования могут служить только данные, поскольку только они объективно отражают производство и в них не отображаются субъективные факторы управления.

На инфологической модели производственного процесса строится стратегический план данных, который затем связывается с функциональной моделью системы. Все системы нижнего

уровня имеют определенную функциональность, они рассчитаны на выполнение определенного класса задач. При этом очень важно, чтобы автоматизация проводилась не программированием отдельных задач управления, а предоставлением средств для решения всего класса задач системой нижнего уровня. Проблема состоит в том, что никто не может определить, сколько и какие именно задачи необходимо решить для выполнения той или иной функции управления. Количество и состав задач имеет сильно изменяющийся характер и не может служить основой создания программного обеспечения автоматизированной системы. Использование т.н. генераторных средств позволит выполнять настраивание системы на решение новых задач изменением описаний обработки для генераторов ввода, отчетов, вычислений и т.д., без дополнительного программирования. Именно это является основным преимуществом настраиваемых систем в сравнении с жестко запрограммированными на определенные задачи подсистемами АСУ.

Функциональные свойства системы экономического анализа

Далее рассмотрим, какие функции работы должны быть обеспечены на рядовом АРМ для выполнения аналитических расчетов. Основой создания соответствующей функциональной модели служат действия, реально выполняемые работниками аппарата управления. Автор статьи разделяет мнение тех, кто считает, что перевод управления на новые, автоматизированные формы должен начинаться с эргономии, с разработки средств автоматизированного проведения привычной работы, а не с переквалифицирования персонала в программисты. Создаваемые системы должны быть максимально дружественными пользователю, при их реализации должны учитываться прежде всего удобства пользователей, даже если это приводит к созданию дополнительного программного обеспечения или приобретению еще некоторых технических средств.

Основной концепцией функциональной модели системы анализа является работа с аналитическими таблицами. Прямо с экрана дисплея в таблицу вводят первичную информацию, затем здесь же вычисляют необходимые промежуточные и производные показатели. Некоторая часть информации (например,

справочные данные или показатели для сравнения) может поступать на рабочее место по каналам связи с других АРМ, но может быть обеспечена и собственными средствами - с других таблиц (таблицы других подразделений, предшествующих периодов, более детального анализа и т.д.), с информационно-справочной системы.

Каждая таблица оснащена уникальным именем, которое должно однозначно определить ее тип (список, свойства и расстановку показателей), период анализа (день, неделя, месяц, квартал, год и т.д.) и конкретное анализируемое подразделение. Таблица состоит из заголовка, шапки, боковины и собственно таблицы в виде полей значений показателей; она может быть дополнена различными атрибутами оформления.

Кроме обработки аналитических таблиц предвидется ведение текстовых документов (отчеты, приказы, списки, письма и др.) и справочной информации в виде картотек.

Функция выдачи вспомогательной информации, как и функция обслуживающих работ является необязательной, но в первом случае наличие функции поможет начинающему пользователю быстрее ознакомиться с системой и ее возможностями, и повысить эффективность ее функционирования во втором.

Основными являются следующие функции:

Т - обработка аналитических таблиц,

С - случайные (одиночные) запросы,

М - моделирование экономических процессов,

Р - обработки текстовой и справочной информации,

О - обслуживание системы,

И - выдача вспомогательной информации.

Изложим содержание функции Т.

Т.I. Основные функции:

- ввод данных в таблицу,
- исправление данных,
- формирование сводных таблиц сложением однотипных таблиц различных подразделений, периодов и т.п.,
- разбиение таблиц на части,
- соединение таблиц и их частей в итоговые (выходные) таблицы,
- временное соединение таблиц для вывода (вывод нескольких таблиц по одной боковине),

- вывод одной или нескольких таблиц на дисплей,
- вывод одной или нескольких таблиц на печать,
- пересылка таблиц по каналам связи.

Т.2. Работа с описаниями типов таблиц:

- редактирование текстов описаний,
- трансляция описаний,
- переименование, копирование и уничтожение описаний.

Т.3. Вспомогательные и обеспечивающие функции:
- идентификация таблиц по имени,
- вывод на дисплей каталога имеющихся таблиц,
- переименование таблиц,
- копирование и уничтожение таблиц,
- поддержание связи с другими подсистемами и системой высшего уровня,
- обработка ошибочных ситуаций.

Уточним функцию С - работу по одиночным запросам на значения каких-то отдельных показателей.

С.I. Основные функции случайных запросов:
- идентификация искомого показателя,
- определение логических условий поиска,
- вывод одного или нескольких значений на дисплей,
- вывод одного или нескольких значений на печать.

С.2. Вспомогательные функции:
- вывод каталога имеющихся таблиц,
- вывод имен и кодов известных показателей,
- вывод имен известных типов таблиц,
- вывод имен известных периодов,
- вывод имен известных подразделений,
- обработка ошибочных ситуаций.

По этим данным пользователь сможет оценить вероятность наличия в локальной базе интересующих его данных и сформулировать семантически правильный запрос.

Функция М - моделирование экономических процессов - дает следующие возможности.

М.I. Использование пакета готовых модулей. Пакет содержит программы, реализующие наиболее известные экономико-математические модели - разыгрывание и симулирование производственных процессов на матрицах, графах, в виде уравнений и т.д.

М.2. Средства для создания новых модулей моделирования:
- транслятор с языка Ассемблера,
- интерпретатор языка ФОРТ,
- интерпретатор языка Бейсик,
- транслятор языка Паскаль и др.

Как и функция С, функция М предназначена в основном для специалистов экономических служб и высшего управляющего персонала.

Функция обработки текстовой и справочной информации Р предназначена для хранения и выдачи любой справочной информации в виде картотек с возможностью перевода их в аналитические таблицы. Также обеспечивается создание, редактирование и тиражирование любых текстовых документов.

Р.1. Обработка справочной информации:
- ввод данных в записи справочной системы,
- исправление записей,
- уничтожение записей,
- поиск записей (выбор по логическим условиям),
- вывод записей на дисплей или печать,
- перевод записей между файлами.

Р.2. Перевод информации из справочной системы в аналитические таблицы:
- определение исходного справочного файла и результатной аналитической таблицы,
- определение логических условий перевода.

Р.3. Обработка текстов:
- командное (строчное) редактирование текстов,
- экранное редактирование текстов,
- копирование и слияние текстов,
- выбор устройства вывода,
- установка размеров и размещения на физическом носителе выходного потока независимо от выводимого текста (статическое форматирование),
- управление выводом специальными символами в тексте (динамическое форматирование).

Режим обслуживания О предназначен в основном для обеспечения работоспособности информационной базы системы, а также для настраивания и перенастраивания системы в ходе эксплуатации.

О.I. Ведение базы данных:

- подготовка (инициализация) носителей данных,
- создание и реорганизация наборов данных,
- копирование, проверка и исправление наборов данных,
- наблюдение за состоянием наборов данных.

О.2. Настраивание и расширение системы:

- средства расширения языка общения с системой,
- средства создания дополнительного программного обеспечения,
- средства стыковки системы с другими системами,
- средства настраивания системы на различные конфигурации ЭВМ,
- средства настраивания системы на различные ЭВМ.

О.3. Допуск к системе и обеспечение секретности данных:

- определение имени и полномочий пользователя,
- ведение учета пользования ресурсами системы,
- модифицирование файлов допуска и учета.

Система выдачи вспомогательной информации И должна обеспечить выдачу инструкций и объяснений, а также поддерживать самообучение начинающих знакомство с системой пользователей.

И.I. Функции управления:

- вывод меню имеющихся информационных страниц,
- выбор искомой страницы,
- движение вперед и назад по страницам,
- возвращение к меню или выход из системы.

И.2. Вывод информационных страниц:

- общее описание системы,
- инструкция оператора,
- инструкция администратора системы (описание архитектуры и реализации системы),
- контрольный пример работы системы,
- инструкция по использованию пакета моделирования,
- описание содержания и порядка вычисления используемых экономических показателей.

Комплект инструкции дополняется инструкциями использования разработанных пользователями средств моделирования.

Заключение

Предложенная в статье функциональная модель является относительно полным описанием требований к системе автоматизированного экономического анализа. Естественно, кое-что может оказаться лишним или недоставать в отношении конкретного предприятия, но по мнению автора, модель такого объема является минимальной для успешной реализации функции экономического анализа на различных АРМ. Следует отметить, что вышеописанная модель не включает, например, обработки графической информации, так как это пока не обеспечивается доступными предприятиям техническими средствами.

Реализованная по этой модели система описана в статье [2]

Литература

I. Кузьминский А.Н., Сопко В.В. Организация бухгалтерского учета и экономического анализа в промышленности: Практ. руководство. - М.: Финансы и статистика, 1984. - 200 с.

2. Рензер А.В. Системы автоматизированного экономического анализа на базе ПОК "АРМ-экономика" // Тр. Таллинск. политехн. ин-та. - 1984. - № 568. - С. 59-69.

A. Renzer

Functional Aspects of Economical Analysis Automation

Abstract

This article deals with economical analysis automation on the production firm level. Strategies of system design are reviewed and functional properties of a suitable automated workplace are described.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

Р.Р. Юргенсон

ОДИН МЕТОД ПРОГРАММИРОВАНИЯ МНОЖЕСТВ НА ЯЗЫКЕ АДА

На языке Ада среди структурных типов данных отсутствует множественный тип, т.е. тип, значениями которого являются множества значений, выбранные из некоторого другого типа, называемого базовым. В то же время существует класс задач, для которых аппаратура множеств позволяет представить алгоритмы более выразительно, а часто и более эффективно, чем это возможно другими средствами языка Ада. Поэтому имеет смысл "добавить" в арсенал средств языка Ада множественный тип и связанные с ним операции.

В данной статье предлагается один способ конструирования для языка Ада множественного типа, совпадающего по своим свойствам с множественным типом языка Паскаль. Вводимый тип представляется как абстрактный тип данных, т.е. наряду с введением нового типа вводятся и операции, применяемые к объектам этого типа. Операции над множествами программируются в виде функций и соединяются в пакет. В некоторых случаях такие функции переопределяют стандартные действия, придавая им новый смысл.

Под языком Ада и Паскаль подразумеваются стандарты этих языков, описанные соответственно в работах [1] и [2].

1. Определение множественного типа и соответствующих объектов

Множественный тип определим как логический массив, где индексным типом является базовый тип:

```
множественный-тип =
"ARRAY""("базовый-тип")" "OF""BOOLEAN".
```

Естественно требовать, чтобы базовый тип был дискретным типом:

базовый-тип = INTEGER | BOOLEAN | CHARACTER|

перечислимый-тип | соответствующие-подтипы.

Множественные переменные объявляются в таком случае обычным образом:

идентификатор {, идентификатор } ":"
множественный-тип ";"

Множественные константы, т.е. постоянные множественного типа или, проще говоря, множества можно строить с помощью агрегатов языка Ада, в которых элементом базового типа, входящим в множество, отвечает значение TRUE, а элементам, не входящим в множество - значение FALSE.

Пустое множество задаётся как агрегат, где все значения равны FALSE.

Оператор присваивания имеет в случае множественного типа следующий вид:

множественная-переменная ":="
выражение-над-множествами ";".

Здесь выражение-над-множествами может содержать арифметические действия над множествами, рассматриваемыми в следующем разделе, и обращения к функциям с множественным значением. В частном случае это выражение может представлять собой множественную переменную или множественную константу.

Примеры.

```
TYPE LETTERS IS CHARACTER RANGE 'A'..'Z';
-- значениями базового типа являются
-- латинские прописные буквы
.
.
.
TYPE LETTERS_SET IS ARRAY (LETTERS) OF BOOLEAN;
-- объявление множественного типа
.
.
.
```

```
VOWELS, FIRSTLETTERS : LETTERS_SET;
-- объявление множественных переменных
.
.
.
VOWELS := ('A', 'I' => TRUE, OTHERS => FALSE);
-- множество, состоящее из букв A и I
.
.
.
FIRSTLETTERS := (OTHERS => FALSE);
-- пустое множество
.
.
.
FIRSTLETTERS := ('A', 'B', 'C' => TRUE, OTHERS => FALSE);
-- множество, состоящее из букв A, B и C
```

Операции над множествами

В Паскале операции над множествами можно разделить на три группы:

- арифметические операции (+, *, -);
- операции сравнения (=, <>, <=, >=);
- операция IN проверки принадлежности элемента к множеству.

Операндами операций первых двух групп являются множественные выражения, обозначим их через A и B. Первым операндом операции IN является скалярная величина (возможный элемент множества).

Рассмотрим нахождение эквивалентов для указанных операций в Аде.

Арифметические операции. Очевидно, что операции A+B и A*B эквивалентны операциям A OR B и A AND B языка Ада соответственно. Конечно, для того, чтобы получить более стильные выражения, эти операции можно переопределить. Например, для получения знака "+" в случае операции сложения можно объявить функцию

```
FUNCTION "+"(A,B : settype) RETURN settype IS
BEGIN
  RETURN A OR B;
END "+";
```

Здесь settype – произвольный множественный тип.

Операция A – B не имеет прямого эквивалента на языке Ада. Поэтому эту операцию надо запрограммировать, например, в виде функции (где basetype – базовый тип множественного типа settype):

```
FUNCTION "-"(A,B : settype) RETURN settype IS
   C : settype;
BEGIN
   FOR I IN basetype LOOP
     IF (A(I) = TRUE) AND (B(I) = FALSE)
       THEN C(I) := TRUE;
       ELSE C(I) := FALSE;
     END IF;
   END LOOP;
   RETURN C;
END "-";
```

Операции сравнения. Операциям Паскаля A = B и A <> B отвечают операции A = B и A = / B языка Ада.

Для операция A <= B и A >= B эквивалентов, выражающихся простыми действиями, нет, но нетрудно составить логические функции, выполняющие эти операции. Эти функции включены в пакет, представленный в следующем разделе.

Операция IN:

Элемент IN A представляется как индексная переменная A (элемент), имеющая значение TRUE, если элемент принадлежит множеству A, и значение FALSE в противном случае.

Пример использования операции IN:

```
IF VOWELS ('O') THEN...;
END IF;
```

2. Пакет для обработки множеств

Чтобы превратить введенные средства в удобный для пользователя механизм, целесообразно соединить их в пакет. Такой пакет можно независимо компилировать и включить в доступную пользователю библиотеку. При этом следует учесть, что пакет не может зависеть от неконкретизированных типов.

Для конкретности, составим пакет для обработки литерных множеств (базовый тип - CHARACTER).

Пакет на языке Ада состоит из спецификации пакета, в которой описываются нужные типы и операции, и тела пакета, где эти операции реализованы, т.е. где имеются полные описания соответствующих подпрограмм.

В нашем случае спецификация пакета (назовем пакет именем CHARSET) должна содержать объявление множественного типа и описания (заголовки) всех перечисленных операций (функций), кроме операции =, =/ и IN. Последними можно пользоваться, опираясь на включенное в спецификацию пакета объявление множественного типа.

Таким образом, спецификация пакета CHARSET будет иметь вид:

```
PACKAGE CHARSET IS
  TYPE SET_TYPE IS ARRAY (CHARACTER)OF BOOLEAN;
  FUNCTION"+"(A,B : SET_TYPE) RETURN SET_TYPE;
  FUNCTION"*"(A,B : SET_TYPE) RETURN SET_TYPE;
  FUNCTION"-"(A,B : SET_TYPE) RETURN SET_TYPE;
  FUNCTION"<="(A,B : SET_TYPE) RETURN BOOLEAN;
  FUNCTION">="(A,B : SET_TYPE) RETURN BOOLEAN;
END CHARSET;
```

Чтобы получить тело пакета CHARSET, следует выписать полные тексты входящих в пакет функций.

```
PACKAGE BODY CHARSET IS
  FUNCTION "+" (A,B : SET_TYPE) RETURN SET_TYPE IS
  BEGIN
    RETURN A OR B;
  END"+"
  FUNCTION "*"(A,B:SET_TYPE) RETURN SET_TYPE IS
  BEGIN
    RETURN A AND B;
  END" ";
  FUNCTION "-"(A,B:SET_TYPE) RETURN SET_TYPE IS
    C : SET_TYPE:
  BEGIN
    FOR I IN CHARACTER LOOP
      IF (A(I) = TRUE) AND (B(I) = FALSE)
```

```
        THEN C(I) := TRUE;
        ELSE C(I) := FALSE;
      END IF;
    END LOOP;
    RETURN C;
END "-";
FUNCTION "<="(A,B : SET_TYPE) RETURN BOOLEAN IS
  CONTAINS : BOOLEAN := TRUE;
BEGIN
  FOR I IN CHARACTER LOOP
    IF (A(I) = TRUE) AND (B(I) = FALSE) THEN
      CONTAINS := FALSE;
      EXIT;
    END IF;
  END LOOP;
  RETURN CONTAINS;
  END "<=";
FUNCTION ">="(A,B : SET_TYPE) RETURN BOOLEAN IS
BEGIN
  RETURN B <= A;
END ">=";
END CHARSET;
```

Составленный пакет имеет определенное ограничение – он может применяться только для литерных множеств. Это ограничение можно устранить путем превращения пакета в генерическую программную единицу, где базовый тип задается как генерический параметр:

```
GENERIC
  TYPE BASE_TYPE IS PRIVATE;
PACKAGE CHARSET IS
  TYPE SET_TYPE IS ARRAY (BASE_TYPE) OF BOOLEAN;
BEGIN
  FUNCTION "=" (A,B : SET_TYPE) RETURN SET_TYPE;
  .
  .
  .
END CHARSET;
```

(Тело пакета не изменяется.)

Такой пакет может быть приспособлен для множеств с произвольными элементами дискретного типа. Какой вариант

пакета предпочесть - генерический или негенерический - это зависит от конкретной ситуации.

Отметим еще, что пакет CHARSET нетрудно дополнить дополнительными операциями, которые могут представлять практический интерес (нахождение дополнения множества относительно базового множества, нахождение количества элементов в множестве, выполнение операции < и т.д.).

Литература

1. The programming language Ada / Reference manual. American National Standards Institute, Inc. ANSI/MIL-STD--1815A-1983 // Lecture Notes in Computer Science. - N 155. - Springer - Verlag, 1983.

2. Addyman A.M., Brewer к. a.o. A draft description of Pascal // Software - Practice and Experience, 1979. - N 9. - P. 381-424.

R. Jürgenson

How to Use Sets in Ada

Abstract

In this paper a technique to program Pascal's sets in Ada is presented. The set declaration and the set operations are collected into an Ada's package.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

Р.Э. Куузик

ГЕНЕРАТОР ГИПОТЕЗ ДЛЯ КАЧЕСТВЕННЫХ ПРИЗНАКОВ

I. Введение

В последние годы наметились некоторые сдвиги в применении методов изучения внутренней структуры исследуемой совокупности объектов. Откажутся от трудоемкой и дорогой классической схемы: статистики, корреляционный, факторный анализы, методы классификации. Вместо них будут применяться более эффективные методы как для первичной, так и для вторичной обработки социологических данных. Таковыми являются детерминационный анализ, ДСМ-метод, методы выделения ядер и др.

Эти методы разделяют исследуемую совокупность на однородные подгруппы $X_i(N_i,M_i)$ ($i=1,\ldots,\kappa$; $1 \leq N_i \leq N$; $i \leq M_i \leq M$; N – число объектов, M – число признаков в совокупности). Такая информация дает возможность исследователю определить, какие объекты и какие признаки в конкретной ситуации поведут себя одинаково, т.е. он получит информацию, которую при помощи классических методов обработки получить невозможно, поскольку они (классические методы) обобщают поведение исследуемой совокупности в целом.

У вышеназванных методов имеются как сильные стороны, так и недостатки. Ниже предлагается их анализ и новый метод генерации гипотез для качественных признаков.

2. Общая ситуация

Детерминационный анализ (далее ДА) [I, 2] позволяет выделить из исследуемой совокупности $X(N,M)$ все реально существующие сочетания значений из M признаков, т.е. все подгруппы $X_i(N_i,M)$, где $i=1,\ldots,\kappa, 1 \leq N_i \leq N$, κ – число реально существующих в совокупности X подгрупп по M признаков каждый.

При ДА социолог должен иметь некоторое представление о поведении материала для комбинирования интересующих его признаков в целях исследования их одновременного поведения.

ДА-метод интерактивный, методика применения его состоит в пошаговом вовлечении новых признаков в дальнейший анализ. Если исследуемых признаков много, то это очень трудоемкая работа. В этом смысле ДА удобен не для генерации гипотез, а для проверки их, а также для выделения новых закономерностей по шаговому принципу.

Метод ДА быстрый, получение однократного ответа требует меньше I,5 минуты (на ЭВМ СМ-4 при средних объемах совокупности, примерно I000 объектов).

ДСМ-метод [3] по сравнению с методом ДА представляет уже попытку автоматизации процесса порождения закономерностей (гипотез). Метод позволяет выделить из исследуемой совокупности реальные сочетания (пары значений, тройки, четверки и т.д.) $X_i(N_i, M_i)$.

При этом предполагается, что исследуемые объекты состоят из двух типов признаков в виде "причина (MI) - следствие (M2)" (M = MI + M2).

При выделении сочетаний (гипотез) применяется следующая методика: в начале выделяются объекты, которые составляют одинаковые пары значений признаков, на основе их выделяются объекты, составляющие тройки значений, затем четверки и т.д. При этом сложность вычислений.

$$S = O(\sum_{i=1}^{K_0} C_N^i),$$

где C_N^i - число комбинаций из N объектов по i.

Такой алгоритм довольно медленный [3] - для определения гипотез по матрице X(I3, 33) на ЭВМ НАЙТРИ-3ØØØ (скорость процессора 50000 оп/сек) требовалось примерно I0 минут. При этом значения признаков были дихотомические (0/I).

Методы выделения ядер (далее МВЯ) для номинальных шкал [4-6] очень быстрые. Самые быстрые МВЯ для выделения одного ядра требуют только два прохода исходной матрицы - сложность вычислений $S = O(MN)$.

Метод позволяет выделить непокрывающие подгруппы элементов (ядра) $X_i(N_i,M_i)$. Но при этом МВЯ предпочитает выделить в ядро элементы с одинаковыми значениями (только единицы или только двойки и т.д.). От применяемой функции взвешивания элементов совокупности $\Pi(X)$ зависит форма ядра (предпочитают численность элементов ядра по N или по M).

Ядро распечатается по объектам, анализ визуальный и поэтому при больших N ($\geqslant 300$) анализ по распечатке не очень удобный.

Ниже предложим алгоритм выделения ядер, который почти свободен от вышеописанных недостатков и при этом не очень медленный - сложность вычислений $S=O(N^2M)$.

3. Определения. Описание алгоритма

Пусть дана конечная совокупность объектов X(N,M), (где N - число объектов, M - число признаков в совокупности), в которой каждый признак j ($j=1,2,\ldots,M$) качественный и имеет градации в промежутке $h_j=1,2,\ldots,\kappa_j$.

Обозначим через $X_H \subseteq X$ подмножество всех объектов имеющих свойство $H=\&_{\ell} h_{\ell}$, $1 \leqslant \ell \leqslant M$.

Определение I

Разрезом называем логическое произведение объектов $x_t \in X_H(N_i,M)$ такое, что $\bigcap_{t=1}^{N_i} X_H = H$, $|X_H| = N_i \leqslant |X| = N$.

Определение 2

Максимальным разрезом Нм называем логическое произведение объектов $x_t \in X_H(N_i,M)$ такое, что для некоторого сочетания $H=\&_j h_j$, $1 \leqslant j \leqslant M$ имеет место соотношение

$$\bigcap_{t=1}^{N_i} X_H = H_M,\ |X_{H_M}| = N_i = |X_H|,\ H_M(=\&_{\ell} h_{\ell}) \supset H,\ 1 \leqslant j < \ell \leqslant M.$$

3.I. Описание алгоритма

Обозначения:

L(P,M) - множество выделяемых ядер

P - число выделяемых ядер

Алгоритм А

А∅: [Начальная установка] $L() \leftarrow 0$, $P \leftarrow 0$

А1: [Цикл по объектам $x_k \in X$. После цикла конец алгоритма]
$k \leftarrow 1,\ldots,N-1$

А2: [Цикл по объектам $x_i \in X$. Определение сочетания. После окончания цикла идти к А1]
$i \leftarrow k+1,\ldots,N$

$H \leftarrow x_i \cap x_k$. Если $H = 0$, то перейти к А2.

А3: [Цикл по h. Сравнение сочетания H с сочетаниями, содержащимися в L]
$h \leftarrow 1,\ldots,P$

Если $H = L(h)$, то идти к А2 (такое сочетание уже существует) $P \leftarrow P+1$, $H \rightarrow L(P)$, идти к А2.

Далее надо определить частоты проявления каждого сочетания.

Для этого можно пользоваться алгоритмом Зобриста [7], который является очень быстрым, но требующим в то же время довольно много памяти ОЗУ. Он формирует список, содержанием которого является перечень объектов для каждой градации каждого признака, содержащего данную градацию данного признака. Далее для определения частоты проявления сочетания не сравнивается сочетание со всеми объектами, а производится битовое (один бит для каждого признака) сравнение на основе информации, содержащейся в списке.

Выделение реально существующих сочетаний - итеративный процесс. Для этого алгоритм А применяется еще раз на матрице L (вместо $X(N,M)$ принимать $L(P,M)$). Итерацию повторить до непоявления новых сочетаний. Можно доказать, что для выделения всех разрезов из исследуемой совокупности X алгоритма А число проходов меньше $\log_2 \max_j |h_j|$, $j = 1,\ldots,M$.

Далее считаем, что объекты $x_i \in X$ являются сочетаниями, полученными в пересечении с собой.

В ходе работы доказаны следующие основные теоремы:

Теорема I. В любой конечной совокупности X число разрезов определено числом различных сочетаний, полученных ло-

гическим произведением попарно выделенных сочетаний из данной совокупности.

Теорема 2. Сочетания, выделенные алгоритмом А, являются разрезами.

Теорема 3. Алгоритм А выделяет все реально существующие в исследуемой совокупности X разрезы.

Теорема 4. Разрез является ядром системы $\Pi(X, \Pi_x)$.

Теорема 5. Число разрезов (Р) меньше числа всевозможных реально существующих в совокупности X сочетаний (РС).

Разница между Р и РС зависит от числа максимальных разрезов.

4. Алгоритм расслоения

В предыдущем пункте мы предлагали алгоритм, где число выделенных ядер может достигать очень большого числа - максимальное число их равняется величине

$$T = \prod_{j=1}^{M} K_j .$$

В действительности это число гораздо меньше. В реальности некая часть ядер повторяется (их исключаем), число выделенных ядер зависит и от числа объектов в совокупности и от числа максимальных разрезов.

Возникает вопрос, как из совокупности ядер выделить самые существенные?

Предлагаем один возможный алгоритм расслоения. Критерием расслоения является максимальная покрываемость элементов исходной совокупности, т.е. исходная совокупность X(N,M) покрыта на 100 %, если выделенными из L ядрами покрыты N × M элементов X.

В работе алгоритма предполагается, что в совокупности ядер все ядра упорядочены в уменьшающемся порядке на основе их частоты проявления и для каждого ядра определено число элементов M1 ($1 \leqslant M1 \leqslant M$) (признаков), содержащихся в нем.

Обозначения

L(P,M) - множество выделенных алгоритмом А ядер

P - число ядер

Q(P) - число элементов (признаков) в ядре ($1 \leqslant Q(P) \leqslant M$)

SAG(P) - частота проявления ядра

A(K,M) - слой ядер

K - конечное число ядер в слое A

SUM(M) - число покрытия для каждого признака j, $1 \leqslant j \leqslant M$, $SUM(j) \leqslant N$

T(M) - вектор последовательности признаков

ℓ - число заполненных клеток в T

$B(M, K_j)$ - вектор градации признаков, содержащихся в слое A

g, f, c, h - вспомогательные переменные

SUM1(M), B1(M,K_j) - вспомогательные массивы. Их содержание аналогично SUM и B, только в отношении одного ядра.

АЛГОРИТМ B

B0: [Начальная установка]
$A() \leftarrow 0$, $SUM() \leftarrow 0$, $T() \leftarrow 0$, $B() \leftarrow 0$, $K \leftarrow 0$, $g \leftarrow 1$, $\ell \leftarrow 1$.

B1: [Цикл через все ядра в L]
$i \leftarrow g, \ldots, P$. При окончании цикла идти к B9.
$SUM1() \leftarrow 0$, $B1() \leftarrow 0$, $h \leftarrow 0$.

B2: [Цикл через вектор последовательности признаков T.]
$f \leftarrow \ell$. Если $\ell > Q(i)$, то $f \leftarrow Q(i)$.

B3: $a \leftarrow 1, \ldots, f$. При окончании цикла идти к B6
Если $T(a) = 0$, то идти к B4, иначе к B5.

B4: [L(i) принадлежит слою A.]
$t \leftarrow a, \ldots, Q(i)$. При окончании цикла $K \leftarrow K+1$, $A(K) \leftarrow L(i)$ идти к B1. $T(t) \leftarrow$ №(L(i,t)) (т.е. номер признака элемента t в ядре L(i)), $SUM(T(t)) \leftarrow SUM(T(t)) + SAG(i)$, $B(T(t), L(i,t)) \leftarrow 1$, $\ell \leftarrow \ell + 1$,
идти к B4.

B5: [Цикл по элементам ядра L(i).]
$b \leftarrow 1, \ldots, Q(i)$. При окончании цикла идти к B1 (в ядре L(i) не нашлось элемента, номер признака которого = T(a)).
$c \leftarrow$ №(L(i,b)). Если $c \neq T(a)$, то идти к B5.
Если $B(c, L(i,b)) = 0$, то $B1(c, L(i,b)) \leftarrow 1$, $h \leftarrow a$, $SUM1(c) \leftarrow SAG(i)$.
Идти к B3.

B6: [Ядро L(i) принадлежит слою A. Суммировать частоты, фиксировать новые градации признаков.]

Если $h \neq 0$, то $b \leftarrow 1,\ldots,\ell$. Если $SUM1(b) \neq 0$, то

$SUM(b) \leftarrow SUM(b) + SUM1(b)$.

$b \leftarrow 1,\ldots, \sum_{j=1}^{M} K_j$. При окончании цикла идти к В7.

Если $B1(b) \neq 0$, то $B(b) \leftarrow 1$.

В7: Если $h = f$ то идти к В8.

$SUM(T(f)) \leftarrow SUM(T(f)) + SAG(i)$.

В8: [Перенос ядра $L(i)$ в слой А.]

$\kappa \leftarrow \kappa + 1$, $L(i) \rightarrow A(\kappa)$, идти к В1.

В9: [Вычисление % покрытия.]

% покрытия $= \left(\sum_{j=1}^{M} SUM(j)\right)/(N \cdot M) \cdot 100\,\%$.

Конец алгоритма.

В результате работы алгоритма В мы получаем последовательность признаков и слой ядер. Этот процесс можно автоматизировать, поместив в В0 $g = 2,3$ и т.д. Получаем различные слои ядер, дающие некую другую последовательность признаков.

На основе выделенных в слое А ядер и соответствующих им частотам проявления можно предположить вариант для упорядочения объектов.

Для этого для каждого объекта в исходной матрице X надо сложить такие частоты проявления ядер $\in A$, которые полностью содержатся в $x_i \in X$: $S(X_i) = S(X_i) + SAG(A(j))$, если $x_i \cap A(j) = A(j)$, $i = 1,\ldots,N$, $j = 1,\ldots,\kappa$.

Полученный таким образом вес $S(X_i)$ позволяет упорядочивать объекты. Если это сделать в уменьшающем порядке, то в верхней части матрицы будут самые типичные объекты в отношении предложенной последовательности признаков, а самые нестандартные объекты в нижней части матрицы.

Заключение

Предложенные в статье алгоритмы несмотря на их трудоемкость работают довольно быстро. Например, реализуемый на ЭВМ СМ-4 (скорость процессора 130000 оп/с) алгоритм А при

исходной матрице X(I200, I4) требовал без распечатки 8000 ядер примерно I5 минут. Работа алгоритма В длится в таких же пределах времени.

Алгоритмы реализованы в ИВЦ Гостелерадио на макроассемблере в ОС РАФОС-ФОРТ.

Литература

I. Чесноков С.В. Детерминационный анализ социально-экономических данных. - М.: Наука, I982. - I68 с.

2. Веселов А.А., Деза В.Н., Подрабинович А.Я. Вычисление характеристик детерминационных связей // Сб. трудов ВНИИСИ. - I980. - № I (методология комплексного исследования социально-экономических систем). -С. 94-99.

3. Забежайло М.И., Финн В.К. Об одном методе автоматического формирования гипотез и его программной реализации // НТИ. - I982. - № 4. - С. 20-26.

4. Муллат И.Э. Экстремальные подсистемы монотонных систем - I // Автоматика и телемеханика. - I976. - № 5. - С. I30-I39.

5. Выханду Л.К. Монотонные системы в анализе данных // Тр. Таллинск. политехн. ин-та. - I98I. - № 5II. - С. 9I-I00.

6. Выханду Л.К. О некоторых методах упорядочения объектов и признаков в системе данных // Тр. Таллинск. политехн. ин-та. - I980. - № 482. - С. 43-50.

7. Ajang G., Taer F. An efficient algorithm for detection of combined occurrences // IPL. - March, 1979. - P. 137-140.

R. Kuusik

Generator Hypotheses for Qualitative Data

Abstract

In this paper a new way for generating hypotheses for finite qualitative data matrix X(N,M) is described.

Corresponding algorithms are described. Theorems are presented, which are proved in the course of working out the algorithms.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

Т.Ф. Лучковский

ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

О пакетах прикладных программ (ППП) пишут и говорят много и это совершенно понятно, так как использования ППП является одним из главных способов перехода к индустриальным методам создания информационных систем. В крупном масштабе ППП можно разделить на две категории - на готовые к эксплуатации пакеты (предназначены главным образом для пользователей информационных систем) и на пакеты, для использования которых нужно писать вызывающие программы (предназначены для программистов-разработчиков информационных систем). Чтобы не повторять уже сказанное другими авторами (напр. [1]), в статье затрагиваются только менее обсужденные моменты неудовлетворительного использования пакетов второй категории при создании новых информационных систем.

I. Причины малого использования пакетов

Субъективные причины. Решая поставленную задачу без применения ППП, программист нуждается только в знании конкретного языка программирования. Для использования ППП он должен еще научиться работать с данной ППП, изучить документацию, считаться с ограничениями и особенностями применяемого ППП. Не каждый программист согласен с такими условиями, и вместо того, чтобы изучать возможности ППП, он ищет причины не использовать пакет.

Если ППП создан во всемирно известной организации и применяется в реализации многих информационных систем, то возражений против данного пакета не так много. Гораздо хуже, когда пакет создан в другом отделе одной организации или в соседней организации. Тогда часто даже не ищут обоснованных причин для критики пакета, подходящими являются и

такие "аргументы", как "Я их знаю, они не способны сделать что-нибудь полезное".

Программистам свойствен непоколебимый оптимизм на счет результатов своей работы. Никакая неудача не может натолкнуть их на мысль, что у них нет необходимых знаний и умений для создания крупных информационных систем. С другой стороны, в неудачах обычно обвиняют других программистов, которые "не умеют писать хорошие и правильные программы". Для некоторых разработчиков применение программного продукта других программистов - это почти признание, что кто-то создал что-то такое, на что не способен он сам.

Возможно, перечисленные причины не являются решающими и они не относятся ко всем программистам, поэтому об этом обычно (вслух) не говорят. Чаще речь идет об объективных причинах, которых тоже не мало.

Пакеты зачастую не готовы к внедрению. Работающие программы - это еще не пакет. Создание программного продукта Брукс [2] считается в три раза более трудоемкой работой, чем просто создание программ. Поэтому нужны стимулы для оформления своих программ в виде ППП, но таких стимулов не слишком много. Более того, часто такое оформление в глазах начальства может казаться излишним, задерживающим реализацию и так запаздывающего проекта.

Оформление ППП - это не только документация о различных возможностях применения пакета. И сами тексты программ должны соответствовать определенным требованиям, программы должны быть более универсальными, надежными и модифицируемыми, чем это необходимо для решения конкретной задачи. С какой стати программист будет брать на себя дополнительную нагрузку? Каким образом он будет распространять свой пакет? Так как он вероятно является одним из ведущих программистов в своей организации, есть ли у него вообще время для этого?

Это были рассуждения с точки зрения разработчика ППП. Теперь со стороны пользователя пакета: ППП необходимо состыковать со своими программами, настроить пакет для решения конкретного применения. Задача часто не тривиальная, здесь требуется не только умение программировать, но и умение собрать программы из готовых частей. Не всякий про-

граммист это умеет (или не хочет уметь, но это было уже названо как субъективная причина).

Программы пакета могут содержать ошибки, и так как они выполняют самые типичные функции - скорость работы этих программ может не удовлетворять пользователя. Для модификации пакета нужны умения сопровождать программный продукт - это одна из самых неизученных областей в процессе создания и сопровождения информационных систем.

Простой пример. Информационная система, собранная из готовых пакетов, выдает диагностические сообщения не только в разном стиле, но даже на разных языках. Пользователь системы недоволен, но кто должен стандартизировать выдаваемые сообщения - автор ППП или автор информационной системы? Работа это не обязательно простая, но по мнению всех программистов, "абсолютно не творческая".

В состав критериев качества программного продукта входят более десяти показателей [3]. Невозможно удовлетворить все критерии одновременно, но пользователь ППП хочет именно этого. Довольно просто прийти к неправильносу выводу "лучше все-таки сделать все с начала до конца своими силами".

2. Как создать пакет?

Самый простой ответ на данный вопрос следующий: ППП должен быть создан таким образом, чтобы у потенциального пользователя не было оснований отказываться от использования данного пакета. Однако осуществить это на практике довольно трудно. Остановимся на некоторых моментах, которые надо учесть при создании ППП.

В первую очередь надо иметь в виду, что пакет создается для других программистов, для разных систем. Необходимо найти разумный компромисс между критериями качества программного продукта, например, между универсальностью и эффективностью программ. Пакет должен иметь определенную степень универсальности - следовательно, логично ожидать, что модули ППП находят применение при реализации разных проектов данной организации (данного отдела, данного автора). Это, во-первых, хороший метод тестирования ППП, и, во-вторых, в данном случае обоснованы затраты на создание

пакета в глазах руководителей проектов. Но если ни автор, ни его коллеги не использовали модули пакета при разных условиях, почему мы требуем (ждем) этого от других? Поэтому наилучший способ убедиться в качестве разработанного пакета - самому быть первым пользователем ППП.

Другой важный момент состоит в том, что ППП для программистов должен состоять из множества как можно более независимых модулей, которые можно использовать как вместе, так и по отдельности для выполнения только определенной функции. Это дает программисту-пользователю пакета возможность по-разному комбинировать модули ППП, заменять при необходимости некоторые из них (например, модули ввода и вывода данных), расширить область применения конкретного пакета. Многие субъективные факторы, упомянутые в предыдущем разделе, в данном случае отпадают.

Литература

I. Морозов В.П. Особенности проектирования систем обработки экономической информации на базе ЕС ЭВМ. - М.: Финансы и статистика, 1982.

2. Брукс Ф.П. Как проектируются и создаются программные комплексы. - М.: Наука, 1979.

3. Боэм Б., Браун Дж., Каспер Х. и др. Характеристика качества программного обеспечения. - М.: Мир, 1981.

T. Luczkowsky

Programming Packages

Abstract

Programming packages are a very useful tool for creating software systems. Some problems with packages and some ways for resolving these problems are discussed in the article.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 517.934

С.М. Геращенко-Тынисмяги, Т.К. Тяхт

КАЧЕСТВЕННОЕ ИССЛЕДОВАНИЕ НЕЛИНЕЙНОЙ САУ ВТОРОГО ПОРЯДКА

Исследуемая система автоматического управления (САУ) относится к классу задач, возникающих при анализе движения космических аппаратов [1, 2].

1. Постановка задачи

Рассмотрим САУ второго порядка вида: $\dot{x} = Ax + bu$ или

$$\begin{cases} \dot{x}_1 = -k_1 \operatorname{sign}\sigma(x), \\ \dot{x}_2 = -bx_2 + k_2 \operatorname{sign}\sigma(x), \end{cases} \qquad (1)$$

где $\sigma(x) = x_1 + x_2 = 0$ – прямая скольжения, $b > 0$, k_1 и k_2 – положительные постоянные,

$$\operatorname{sign}\sigma(x) = \begin{cases} 1, & \text{при } \sigma(x) > 0, \\ 0, & \text{при } \sigma(x) = 0, \\ -1, & \text{при } \sigma(x) < 0. \end{cases}$$

САУ (1) обладает следующими особенностями: а) невозможность замера x_1, а только (x_1+x_2), б) коэффициенты усиления входят в систему с разными знаками. При $k_1=0$ получается из (1) система с положительной обратной связью, а при $k_2=0$ – система с отрицательной обратной связью. В процессе регулирования при $k_1 \neq 0$ и $k_2 \neq 0$ неизбежно наступают такие моменты, что величина (x_1+x_2) мало отличается от величины x_1, и тогда система (1) аналогична системе с отрицательной обратной связью, в другие моменты времени величина (x_1+x_2) мало отличается от величины x_2, и тогда система (1) аналогична системе с положительной обратной связью.

Задачей управления такой нелинейной САУ является, грубо говоря, примирить противоположные интересы составляющих системы (1).

Цель исследования – установить общую качественную картину процесса управления САУ (I) и условия асимптотической устойчивости движения [2, 4].

2. Исследование системы

В системе (I) скользящий режим [3] возникает на прямой $\sigma(x)=x_1+x_2=0$. Рассмотрим системы дифференциальных уравнений, соответствующие системе (I) при $\sigma(x)>0$:

$$\begin{cases} \dot{x}_1 = -k_1, \\ \dot{x}_2 = -bx_2+k_2, \end{cases} \qquad (2^+)$$

и при $\sigma(x)<0$:

$$\begin{cases} \dot{x}_1 = k_1, \\ \dot{x}_2 = -bx_2-k_2. \end{cases} \qquad (2^-)$$

Возьмем производную по времени в силу системы (I) от функции $\sigma(x)$

$$\frac{d\sigma(x)}{dt} = -(k_1-k_2)\operatorname{sign}\sigma(x)-bx_2. \qquad (2)$$

Из выражения (2) видно, что нужно проанализировать несколько случаев.

I. $k_1 > k_2$.

На отрезке $\sigma=0, |x_2|<\frac{k_1-k_2}{b}$ существует скользящий режим в (I).

а) $k_1 > 2k_2$.

$$\frac{dx_2}{dx_1} = \frac{bx_2-k_2}{k_1} \quad \text{или} \quad \frac{dx_2}{bx_2-k_2} = \frac{dx_1}{k_1}.$$

и решение получается в виде

$$x_2 = \frac{k_2}{b} + \frac{1}{b}\exp\left(\frac{bx_1}{k_1}+c\right) \quad \text{при} \quad x_2 > \frac{k_2}{b}$$

и

$$x_2 = \frac{k_2}{b} - \frac{1}{b}\exp\left(\frac{bx_1}{k_1}+c\right) \quad \text{при} \quad x_2 < \frac{k_2}{b},$$

если

$$x_1 = x_2 = -k_2/b, \text{ то } \exp\left(c-\frac{k_2}{k_1}\right) = 2k_2.$$

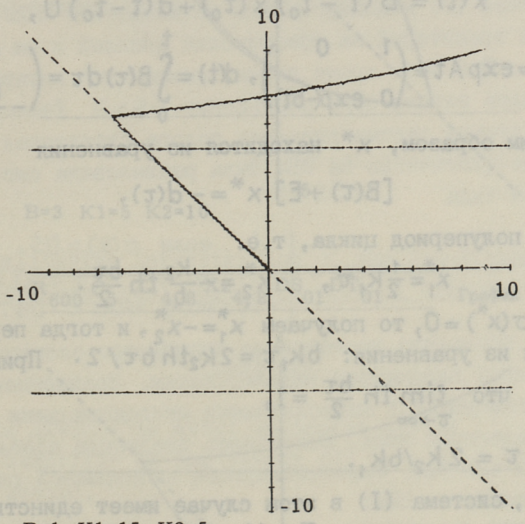
В этом случае скользящий режим описывается уравнениями

$$\dot{x}_1 = \frac{bk_1}{k_1 - k_2} x_2$$

$$\dot{x}_2 = -\frac{bk_1}{k_1 - k_2} x_2$$

т.е. $\dot{x}_1 + \dot{x}_2 = 0$ (см. график I).

б) $k_1 > k_2$, но $\frac{k_1 - k_2}{b} < \frac{k_2}{b}$.



B=1 K1=15 K2=5

R_{500} C_{10} $X1_9$ $X2_9$ DS_{01} DT_{01} График 1.

Получаем качественную картину движения, аналогично случаю I а).

II. $k_1 < k_2$.

В этом случае скольжения не существует, на отрезке прямой

$$\sigma(x) = 0, \quad |x_2| < \frac{k_2 - k_1}{b}$$

происходит отталкивание траекторий. Касательная к траектории в точке $(-\frac{k_2}{b}; \frac{k_2}{b})$ имеет вид

$$\frac{dx_2}{dx_1} = \begin{cases} \dfrac{-bx_2 + k_2}{-k_1} = -\dfrac{2k_2}{k_1}, & \sigma^+ = +0, \\ \dfrac{-bx_2 - k_2}{k_1} = -0, & \sigma^- = -0, \end{cases}$$

где $2k_2/k_1 > 1$, т.к. $k_1 < 2k_2$, $k_2 > 0$. В системе (I) возникает предельный устойчивый цикл (единственный). Покажем это. При $U = const$ по формуле Коши можно записать решение в виде

$$x(t) = B(t - t_0)\,\dot{x}(t_0) + d(t - t_0)\,U,$$

где $B(t) = \exp At = \begin{pmatrix} 1 & 0 \\ 0 & \exp(-bt) \end{pmatrix}$, $d(t) = \int_0^t B(\tau)\,d\tau = \begin{pmatrix} -k_1 t \\ -\dfrac{k_2}{b} + \dfrac{k_2}{b} e^{bt} \end{pmatrix}$.

Таким образом, x^* находится из уравнения

$$[B(\tau) + E]\,x^* = -d(\tau), \tag{3}$$

где τ – полупериод цикла, т.е.

$$x_1^* = \frac{1}{2} k_1 \tau, \qquad x_2^* = -\frac{k_2}{b} \operatorname{th} \frac{b\tau}{2}.$$

Так как $\sigma(x^*) = 0$, то получаем $x_1^* = -x_2^*$, и тогда период цикла находится из уравнения: $bk_1\tau = 2k_2 \operatorname{th} b\tau/2$. Принимая во внимание, что $\lim\limits_{\tau \to \infty} \operatorname{th} \frac{b\tau}{2} = 1$,

получаем $\tau = 2k_2/bk_1$.

Итак, система (I) в этом случае имеет единственный предельный цикл с периодом $T = 4k_2/bk_1$ (см. график 2).

3. Исследование на устойчивость движения системы (I)

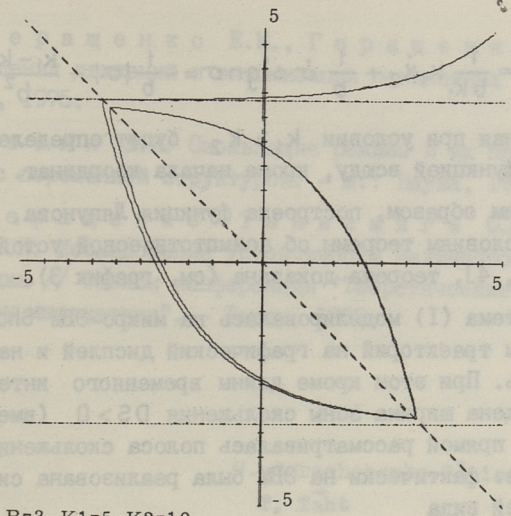
Теорема. Система автоматического управления (I) асимптотически устойчива в целом относительно начала координат, если $k_1 \geqslant k_2$.

Доказательство. Рассмотрим функцию

$$V(x_1, x_2) = \frac{1}{2bk_1} x_1^2 + \frac{1}{b^2} |\sigma(x)|. \tag{4}$$

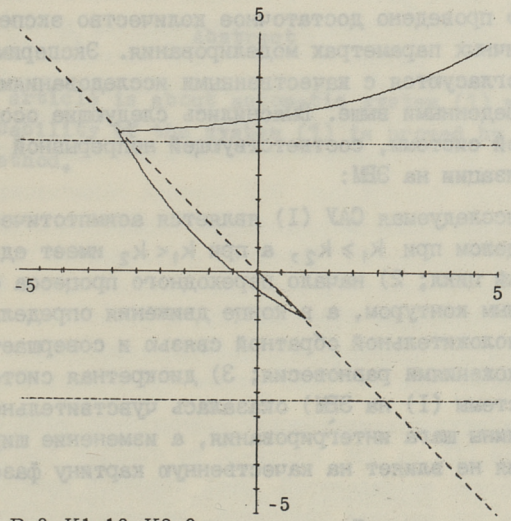
Она является бесконечно большой, определенно-положительной всюду в E^2 и обращается в нуль только в начале координат.

Возьмем производную по времени в силу системы (I)



B=3 K1=5 K2=10

$R_{600}\ C_5\ X1_{4,8}\ X2_{4,8}\ DS_{01}\ DT_{01}$ **График 2**



B=3 K1=10 K2=8

$R_{200}\ C_5\ X1_{4,8}\ X2_{4,8}\ DS_{01}\ DT_{01}$ **График 3**

$$\dot{V} = \frac{1}{b k_1} x_1 \dot{x}_2 + \frac{1}{b} \dot{\sigma} \operatorname{sign}\sigma = -\frac{1}{b}|\sigma| - \frac{k_1 - k_2}{b^2},$$

производная при условии $k_1 \geq k_2$ будет определенно-отрицательной функцией всюду, кроме начала координат.

Таким образом, построена функция Ляпунова, удовлетворяющая условиям теоремы об асимптотической устойчивости в целом [2, 4], теорема доказана (см. график 3).

Система (I) моделировалась на микро-ЭВМ "Sharp MZ-800" с выводом траекторий на графический дисплей и на графопостроитель. При этом кроме длины временного интервала $DT > 0$ была введена ширина зоны скольжения $DS > 0$ (вместо скольжения по прямой рассматривалась полоса скольжения шириной DS), т.е. фактически на ЭВМ была реализована система (I) с функцией вида

$$\operatorname{sign}\sigma(x) = \begin{cases} 1, & \text{если } \sigma(x) > DS, \\ 0, & \text{если } -DS \leq \sigma(x) \leq DS, \\ -1, & \text{если } \sigma(x) < -DS. \end{cases}$$

Было проведено достаточное количество экспериментов при различных параметрах моделирования. Эксперименты полностью согласуются с качественными исследованиями системы (I), приведенными выше. Выяснились следующие особенности дискретной системы, соответствующей непрерывной системе (I) при реализации на ЭВМ:

I) исследуемая САУ (I) является асимптотически устойчивой в целом при $k_1 \geq k_2$, а при $k_1 < k_2$ имеет единственный предельный цикл; 2) начало переходного процесса определяется основным контуром, а в конце движения определяется контуром с положительной обратной связью и совершает колебания между положениями равновесия; 3) дискретная система (реализация системы (I) на ЭВМ) оказалась чувствительной к изменению величины шага интегрирования, а изменение ширины полосы скольжения не влияет на качественную картину фазовых траекторий.

Литература

I. Охоцимский Д.Е., Голубев Ю.Ф., Сихарулидзе Ю.Г. Алгоритмы управления космическим аппаратом при входе в атмосферу. – М.: Наука, 1975.

2. Геращенко Е.И., Геращенко С.М. Метод разделения движений и оптимизация нелинейных систем. – М.: Наука, 1975.

3. Уткин В.И. Скользящие режимы и их применение в системах с переменной структурой. – М.: Наука, 1974.

4. Геращенко-Тынисмяги С.М., Тяхт Т.К. Исследование устойчивости нелинейной САУ второго порядка // Тезисы конференции "Теоретические и прикладные вопросы математики". – Тарту, 1985.

S. Gerashchenko-Tõnismägi,
T. Täht

Qualitative Investigation of the Non-Linear Two-Dimensional Automatic Control System

Abstract

This article is about automatic system (1) with relay control. Stability of the system (1) is proved by Lyapunov-function method.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 528.94

С.М. Тынисмяги, Х.А. Тынисмяги

СХЕМЫ - МОДЕЛИ ВЗАИМОСВЯЗЕЙ ПРИЗНАКОВ И ФАКТОРОВ ПРИ ИССЛЕДОВАНИИ ПРИЧИН ПРЕСТУПЛЕНИЙ

Преступность, как массовое социальное явление, представляет собой совокупность отдельных преступлений и подчиняется закону больших чисел, что позволяет использовать в исследовании математические методы. "Преступления, - отмечал К. Маркс, - взятые в большом масштабе, обнаруживают по своему числу и по своей классификации такую же закономерность, как явления природы..."

Цель нашей работы - с помощью математических методов выяснить криминологическую характеристику изнасилований по статистическим данным Эст.ССР. Для исследования в качестве эмпирической базы были выборочно взяты уголовные дела с обвинительным приговором по изнасилованиям, совершенным за длительный период времени в Эст.ССР [I, 2].

Анализу подверглось 832 эпизода изнасилований с 705 преступниками и 535 потерпевшими, каждое из которых описывалось I6I признаками. Разность в количестве преступников и потерпевших объясняется тем, что большая часть (55 %) преступлений совершена группами лиц, а разность в количестве эпизодов и преступников - одно лицо совершило несколько преступлений, все это было учтено при кодировании данных.

Обработка статистических данных была осуществлена в ВЦ ТГУ Эст.ССР на ЭВМ Минск-32 по типовым программам социологических исследований. Весь числовой материал был разбит на три массива признаков, характеризующих:

I) преступника,

2) потерпевшую,

3) само преступление.

I - массив "преступник" охарактеризован следующими группами признаков (в скобках - номер признака по кодированию):

I. Социально-культурный статус (3, 87, 89, 91, 93, 95 - 97, 102, 110, 190, 199, 200).

2. Характеристика родителей (92, 105, 112, 126).

3. Жизненный путь (86, 94, 99, 100, 121, 124, 125. 127, 134, 135, 144, 166).

4. Характеристика самого преступника (27, 88,90, 98, 101, 104, 106-108, 111, 113, 114, 146, 147, 187).

5. Психосоматическая характеристика (119, 120, 122 - 124, 128-133, 136, 137-139, 145, 186).

6. Доходы преступника (115-119).

6. Ранее совершенные преступления (157-185, 188, 206-213).

7. Характеристика отношения к преступлению после его совершения (признание вины) (62-68).

П - массив "потерпевшая" охарактеризован:

I. Социально-культурный статус (5, 69, 74, 76, 191, 199, 200, 201).

2. Характеристика самой потерпевшей (75, 77, 84, 204-205).

3. Ее состояние во время преступления (75, 78, 82, 189).

Ш - массив "преступление" охарактеризован:

I. Общая характеристика преступления (32, 45, 47, 55-61, 83, 85, 146-156).

2. Характеристика времени и места (15-20, 22, 23, 28-30, 192-198).

3. Характеристика группы преступников (21, 24-26, 31).

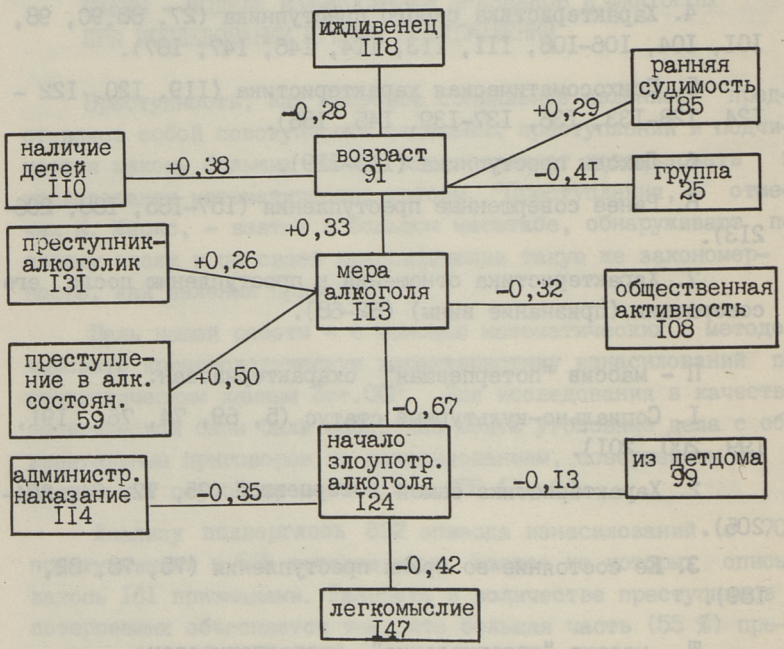
4. Мотив преступления (48-54), повод (140-143).

5. Мера и вид наказания (7).

С помощью одномерного анализа каждый массив сократили до одной пятой, оставив наиболее существенные признаки. По

каждому оставшемуся массиву были вычислены коэффициенты линейной корреляции r_{xy} между всеми признаками, что позволило вновь сжать массив данных, т.е. оставить те признаки, которые имели $|r_{xy}| > 0{,}15$. По методу нахождения пути максимальной корреляции [3] были построены схемы – модели I, 2, 3 взаимосвязей признаков.

СХЕМА – МОДЕЛЬ I

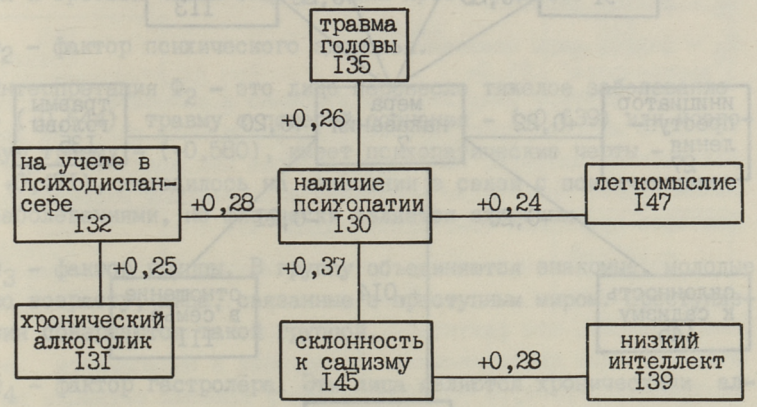


Признак II3 – алкогольное состояние преступника.

Итоги по этой схеме-модели: I) чем моложе преступник, тем сильнее его тяготение к группе; 2) для преступника характерно употребление алкоголя вообще, а не столь важно в каком состоянии он был в момент преступления; 3) важен стаж употребления алкоголя, т.к. обычно преступление совершается на 3–4-й год злоупотребления алкоголя, а не в I–2-й год (наступает деградация личности).

Так как массивы данных продолжали оставаться громоздкими, то применили центроидный метод факторного анализа, разработанный Л. Терстоуном, по каждому массиву отдельно. Перед интерпретацией факторных нагрузок осуществили опера-

СХЕМА - МОДЕЛЬ 2



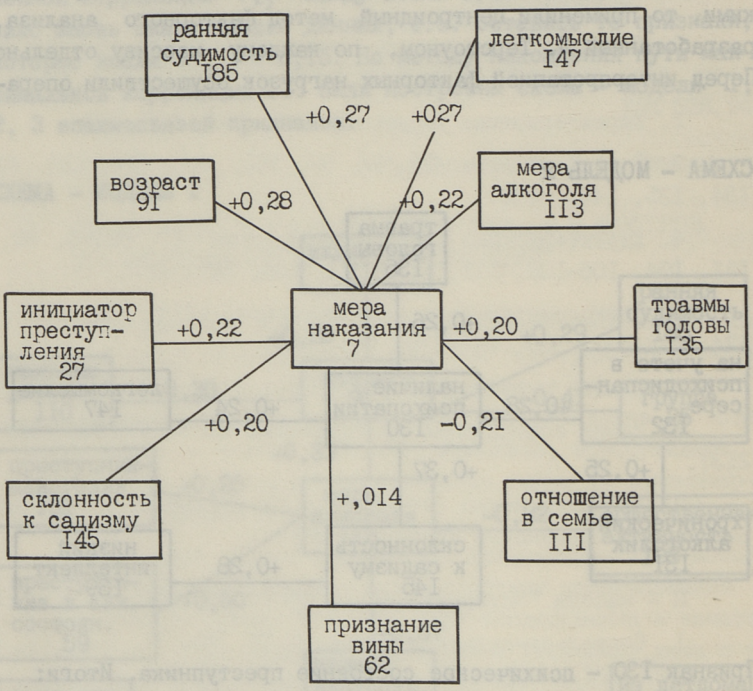
Признак I30 - психическое состояние преступника. Итоги:

I) психопатия занимает центральное место в характеристике преступника. Такими они не рождаются, а становятся под действием окружающей среды, алкоголя, болезней, следовательно, его (преступника) нужно не только наказывать, но и лечить.

По данным из одномерного анализа имеем, что среди осужденных преступников-насильников в Эстонии находится на учете в психодиспансере - 5,2 %, с низким интеллектом - 9,2 % и являются хроническими алкоголиками - 7,7 %.

цию вращения факторов, благодаря которой удалось получить максимальное разграничение факторов путем разделения совокупности параметров на несколько групп, каждая из которых концентрируется вокруг определенного фактора, а за критерий оптимальной факторной структуры был использован принцип "простой структуры" Л. Терстоуна [4]. В результате были

СХЕМА - МОДЕЛЬ 3



Признак 7 - мера наказания

Мера наказания тесно связана с возрастом, характером, мерой злоупотребления алкоголя и преступной биографией преступника.

Мало учитывается судами признание вины преступника, которое по закону является смягчающим обстоятельством.

извлечены по каждому массиву до семи факторов, которым дана содержательная интерпретация. Наиболее существенные из них приведем по массиву "преступник":

Φ_I - фактор алкоголизма, в него вошли -

употребление алкоголя	+0,742,
с какого возраста пьет	-0,737,
прежнее административное наказание	+0,562,
характеристика с места работы	-0,56I
грубость и легкомыслие	+0,534,

общественная активность -0,522,
состояние опьянения в момент преступления +0,434

Интерпретация Φ_I следующая: это лицо употребляет много алкоголя, начал пить, будучи несовершеннолетним, ранее был наказан в административном порядке, его характеристика отрицательна, груб, легкомыслен и общественно не активен, совершил преступление в алкогольном опьянении (это относится к третьей части общего числа преступников).

Φ_2 - фактор психического здоровья.

Интерпретация Φ_2 - это лицо перенесло тяжелое заболевание - (+0,644), травму с потерей сознания - (+0,639) или мозговую травму - (+0,580), имеет психопатические черты - (+0,471), находилось на излечении в связи с психическими заболеваниями, но физически является здоровым.

Φ_3 - фактор группы. В группу объединяются знакомые, молодые по возрасту, лица, связанные с преступным миром. Преступления совершаются такой группой.

Φ_4 - фактор гастролёра. Эти лица являются хроническими алкоголиками с садистскими чертами характера, доход получают преступным путем, не имеют постоянного места жительства, ведут бродячий паразитический образ жизни, в их семьях выпивают.

Выяснилось, что национальность преступника и потерпевшей в совершении преступления не играет никакой роли.

По массиву II - "потерпевшая":

Φ_I - фактор молодости.

Φ_2 - фактор меры опьянения преступника в момент совершения преступления.

Φ_3 - фактор взаимоотношений с преступником.

Φ_4 - фактор национальности (эстонка - 0,944, другие национальности - 0,943).

Φ_5 - фактор группового посягательства.

Φ_6 - фактор незанятости потерпевшей (не работает, не учится).

Φ_7 - фактор легкомыслия.

По массиву III - "преступление":

Φ_1 - фактор времени преступления.

Φ_2 - фактор группового преступления.

Φ_3 - фактор города.

Φ_4 - фактор покушения.

Φ_5 - фактор деревни.

Φ_6 - фактор меры наказания.

Эти факторы были введены вторично в качестве новых признаков в массив "преступления", для них была получена корреляционная матрица, на основе которой был построен путь максимальной корреляции и сделаны схемы-модели взаимосвязи факторов-признаков (см. схема-модель 4).

Содержательный анализ пути максимальной корреляции между признаками, характеризующими преступника и преступление, показал, что фактически преступник и преступление составляют одно целое явление. Преступник в зависимости от внутренней структуры своей личности и внешней ситуации выбирает способ совершения преступления, объект посягательств и т.д.

Фактор времени совершения преступления связан с факторами возраста потерпевшей, группы и алкоголизма очень тесно.

Фактор меры употребления алкоголя связан с факторами меры наказания, деятельности и фактором деревни.

Таким образом, некоторая модификация в использовании методов факторного анализа позволила выявить наиболее существенные признаки и их связи, т.е. осуществить в некотором смысле оптимальный выбор признаков, наиболее ярко характеризующих преступника и преступление.

В ы в о д

Центральное место занимает фактор времени по годам совершения преступления, имеющий значительную связь с факторами алкоголизма, группы и возраста потерпевшей.

Выяснилось, что в нашей республике преступность меняется в течение короткого периода времени, есть разница в преступности 1960-х и 1970-х годов. Насильники стали моложе

по возрасту, более подвижными (в смысле смены места жительства), больше употребляют алкоголь, нападения совершают чаще с хулиганскими побуждениями.

СХЕМА – МОДЕЛЬ 4

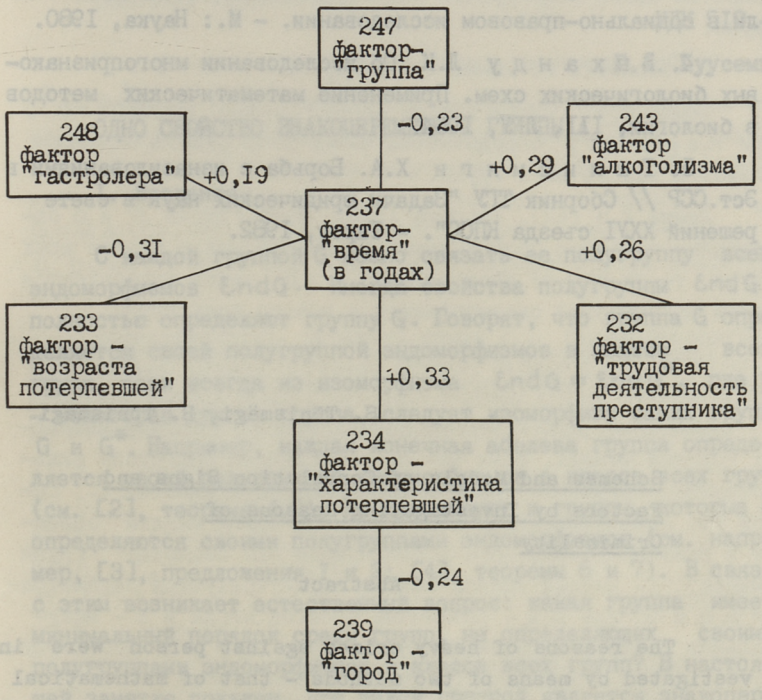


Схема – модель взаимосвязи факторов, где на центральном месте находится фактор "время" (в какие годы совершено преступление).

На втором месте находится фактор возраста потерпевшей, с которым связаны факторы группы, места совершения преступления, меры наказания преступника и фактор его деятельности.

На основе изложенного сделаны рекомендации для предотвращения и профилактики данного вида преступлений [5].

Литература

I. Тынисмяги Х.А. Об особо тяжких последствиях изнасилования // Сборник тезисов Республиканской научной

конференции. Латв. ГУ. - Рига, 1975. 23 - 25 апреля.

2. Тынисмяги Х.А. Недостатки полового воспитания молодежи как криминогенный фактор // Сборник "Правовое воспитание сегодня",-Таллин, 1981 (на эстонском языке).

3. Гаврилов О.А. Математические методы и модели в социально-правовом исследовании. - М.: Наука, 1980.

4. Выханду Л.К. Об исследовании многопризнаковых биологических схем. Применение математических методов в биологии, III, ЛГУ, 1964.

5. Тынисмяги Х.А. Борьба с изнасилованиями в Эст.ССР // Сборник ТГУ "Задачи юридических наук в свете решений XXУI съезда КПСС". - Тарту, 1982.

S. Tõnismägi, H. Tõnismägi

Schemes and Models of Correlation Signs and Factors by Investigating Reasons of Criminality

Abstract

The reasons of heavy crimes against person were investigated by means of two methods - that of mathematical statistics and a modification of the method of factor analysis. The statistic row includes criminal cases in Estonia.

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 519.4

П.А. Пуусемп

ОДНО СВОЙСТВО ЗНАКОПЕРЕМЕННОЙ ГРУППЫ A_4

I. Введение

С каждой группой G можно связать ее полугруппу всех эндоморфизмов $\mathcal{E}ndG$. Иногда свойства полугруппы $\mathcal{E}ndG$ полностью определяют группу G. Говорят, что группа G определяется своей полугруппой эндоморфизмов в классе всех групп, если всегда из изоморфизма $\mathcal{E}ndG \simeq \mathcal{E}ndG^*$, где G^* – некоторая другая группа, следует изоморфизм самих групп G и G^*. Например, каждая конечная абелева группа определяется своей полугруппой эндоморфизмов в классе всех групп (см. [2], теорема 4.2). Но существуют и группы, которые не определяются своими полугруппами эндоморфизмов (см. например, [3], предложения I и 2; [4], теоремы 6 и 7). В связи с этим возникает естественный вопрос: какая группа имеет минимальный порядок среди групп, не определяющих своими полугруппами эндоморфизмов в классе всех групп? В настоящей заметке покажем, что такой группой является знакопеременная группа A_4, имеющая порядок I2.

2. Сведения об определяемости некоторых групп их полугруппами эндоморфизмов

Приведем сначала примеры групп, которые определяются своими полугруппами эндоморфизмов.

Лемма 2.I. ([2], теорема 4.2). Каждая конечная абелева группа определяется своей полугруппой эндоморфизмов в классе всех групп.

Лемма 2.2. ([5], следствие I). Группа кватернионов $Q = \langle a, b \mid a^4=1, b^2=a^2, aba=b \rangle$ определяется своей полугруппой эндоморфизмов в классе всех групп.

Лемма 2.3 ([3], теорема 27). Каждая конечная группа G, являющаяся полупрямым произведением $G = \langle a \rangle \lambda \langle b \rangle$ двух своих циклических подгрупп $\langle a \rangle$ и $\langle b \rangle$, $\langle a \rangle \simeq C_{p^n}$ (циклическая группа порядка p^n), p - произвольное простое число, определяется своей полугруппой эндоморфизмов в классе всех групп.

В частности из леммы 2.3 следует

Лемма 2.4. Группа диэдра $D_n = \langle a, b \mid b^2 = a^n = 1, b^{-1}ab = a^{-1} \rangle$ при $n \in \{3; 4; 5\}$ определяется своей полугруппой эндоморфизмов в классе всех групп.

Лемма 2.5 ([2], теорема I.I3). Если группы A и B определяются своими полугруппами эндоморфизмов в классе всех групп, то их прямое произведение $A \times B$ также определяется своей полугруппой эндоморфизмов в классе всех групп.

Приведем теперь один пример группы, которая не определяется своей полугруппой эндоморфизмов.

Напомним, что конечную ненильпотентную группу называют группой Шмидта, если все ее собственные подгруппы нильпотентны. Строение групп Шмидта описано в работах Л. Редей [9, I0]. Полугруппы эндоморфизмов групп Шмидта описаны в работах [6–8]. Перечислим нужные нам свойства групп Шмидта: I) каждая группа Шмидта характеризуется параметрами p, q и ν, где p, q - различные простые числа и ν - натуральное число; 2) заданным параметрам p, q и ν соответствует, вообще говоря, много групп Шмидта; среди них существует в точности изоморфизма только одна группа G_{max} максимального порядка и одна группа G_{min} минимального порядка; 3) $|G_{min}| = q^{\nu} p^{u}$, где u - наименьшее натуральное число, для которого $p^{u} \equiv 1 (\mathrm{mod}\, q)$; 4) если число u нечетно, то $G_{min} = G_{max}$ и заданным параметрам соответствует только одна группа Шмидта; 5) если число u четно, $u = 2t$, то $|G_{max}| = q^{\nu} p^{3t}$ и все группы Шмидта с параметрами p, q и ν получаются как факторгруппы G_{max}/B, где B - произвольная подгруппа второго коммутанта G''_{max} группы G_{max} (естественно, $G_{min} = G_{max}/G''_{max}$).

Лемма 2.6 ([6], теорема 3.7). Полугруппы всех эндоморфизмов групп G_{min} и G_{max} изоморфны.

Лемма 2.7 ([8], теорема 3.9). Если полугруппа всех эндоморфизмов некоторой группы G^* изоморфна полугруппе всех эндоморфизмов группы Шмидта G с параметрами p, q и ν, то группа G^* является также группой Шмидта с параметрами p, q и ν.

Основная теорема

Сформулируем основной результат настоящей работы.

Теорема. Среди тех групп, которые не определяются своими полугруппами эндоморфизмов в классе всех групп, минимальный порядок имеет знакопеременная группа A_4. Группы G, для которых $\mathcal{E}ndG \simeq \mathcal{E}ndA_4$, исчерпываются группами $G = A_4$ и $G = Q \lambda \langle c \rangle$, где $\langle c \rangle$ – циклическая группа порядка 3 и $c^{-1}ac = b$, $c^{-1}bc = ab$ (a, b – образующие группы кватернионов Q).

Доказательство. Согласно лемме 2.1 будем рассматривать только некоммутативные группы. В книге [1, с. 194], приведен полный список некоммутативных групп порядка < 32. Из этого списка перечислим здесь все группы до порядка 12:

а) D_3 – порядка 6;
б) D_4, Q – порядка 8;
в) D_5 – порядка 10;
г) $D_6 = C_2 \times D_3$ – порядка 12;
д) $\langle a \rangle \lambda \langle b \rangle = \langle a, b \mid b^4 = a^3 = 1,\ b^{-1}ab = a^{-1} \rangle$ – порядка 12;
е) A_4 – порядка 12.

По леммам 2.2–2.5 группы а)–д) определяются своими полугруппами эндоморфизмов в классе всех групп. Остается исследовать случай группы A_4.

Рассмотрим группы Шмидта с параметрами $p=2, q=3$ и $\nu=1$. В таком случае число u, указанное в описании групп Шмидта, равняется двум: $u = 2t = 2, t = 1$. Тогда $|G_{min}| = 12$, $|G_{max}| = 24$ и G_{min}, G_{max} – единственные группы Шмидта с параметрами $p=2, q=3$ и $\nu=1$. Следовательно, группа G_{min} совпадает с группой A_4. По предложению 3 из работы [10] $G_{max} = Q \lambda \langle c \rangle$, где $\langle c \rangle$ – циклическая группа порядка 3, $c^{-1}ac = b$, $c^{-1}bc = ab$ и a, b – образующие группы кватернионов. Теперь все утверждения теоремы следуют из лемм 2.6 и 2.7. Теорема доказана.

Литература

I. Коксетер Г.С.М., Мозер У.О. Дж. Порождающие элементы и определяющие соотношения дискретных групп. -М., 1980.

2. Пуусемп П. Идемпотенты полугрупп эндоморфизмов групп // Уч. зап. Тартуск. ун-та. - 1975. - № 366. - С. 76-104.

3. Пуусемп П. Об определяемости полупрямого произведения циклических групп своей полугруппой эндоморфизмов // Тр. Таллинск. политехн. ин-та. - 1980. - № 482. - С. 131-148.

4. Пуусемп П. Неизоморфные метабелевы группы с изоморфными полугруппами эндоморфизмов // Тр. Таллинск. политехн. ин-та. - 1981. - № 511. - С. 129-140.

5. Пуусемп П. Полугруппы эндоморфизмов обобщенных групп кватернионов // Уч. зап. Тартуск. ун-та. - 1976. - № 390. - С. 84-103.

6. Пуусемп П. Полугруппы эндоморфизмов групп Шмидта // Тр. Таллинск. политехн. ин-та. - 1983. - № 554. - С. 155-164.

7. Пуусемп П. Об автоморфизмах групп Шмидта // Тр. Таллинск. политехн. ин-та. - 1983. - № 554. - С. 165-168.

8. Пуусемп П. Абстрактная характеристика групп Шмидта по их полугруппам эндоморфизмов // Тр. Таллинск. политехн. ин-та. - 1984. - 568. - С. 91.

9. Redei L. Das "schiefe Produkt" in der Gruppentheorie... // Comm. Math. Helv. - 1947. - 20. - S. 225-264.

10. Redei L. Die endlichen einstufig nichtnilpotenten Gruppen // Publ. Math. - 1956. - 4. - S. 303-324.

P. Puusemp

A Property of the Alternating Group A_4

Abstract

All the endomorphisms of an arbitrary group G form a semigroup EndG. Let $\mathfrak{Y}$ be the class of all groups and G a fixed group. If from the isomorphism of semigroups of all endomorphisms of groups G and $H \in \mathfrak{Y}$ follows the isomorphism of groups G and H, then we say that the group G is determined by its semigroup of endomorphisms in the class of all groups.

Let $\mathcal{K}$ denote the class of all groups which are determined by their semigroups of endomorphisms in the class of all groups. In this paper the following result is proved: the alternating group A_4 is the group of minimal order in the class $\mathfrak{Y} \setminus \mathcal{K}$.

Содержание

№ 645

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

ОБРАБОТКА ДАННЫХ, ПОСТРОЕНИЕ ТРАНСЛЯТОРОВ ВОПРОСЫ ПРОГРАММИРОВАНИЯ

ТРУДЫ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА LXII

УДК 681.03

Раскрытие понятий для пользователя проблемно-ориентированной информационной системы. Тепанди Я.Я. – Труды Таллинского политехнического института, 1987, № 645, с. 3–16.

Пользователь проблемно-ориентированной системы нуждается в четких и ясных определениях понятий предметной области. Способы определения этих понятий должны быть максимально приближенными к соответствующим принципам в естественных языках.

На материале толковых словарей русского и английского языков проанализированы следующие способы определения понятий: введение новой роли в есть-понятие, заполнение явно указанной в есть-понятии роли, замена заполненной роли другой, эвристические правила формирования понятий, определение без есть-понятия, иллюстрированный материал в определениях.

Рисунков – 3, библ. наименований – 12.

УДК 681.03

Загрузка и пополнение базы знаний в диагностической экспертной системе. Тепанди Я.Я. – Труды Таллинского политехнического института, 1987, № 645, с. 17–24.

Рассматривается диагностическая экспертная система ТООТС, база знаний (БЗ) которой имеет структуру расширенного бинарного дерева. Для манипулирования БЗ можно пользоваться режимами поиска решения, ввода новых решений в ходе диалога, загрузки БЗ в ходе диалога, загрузки БЗ с

помощью редактора текста. Приводится алгоритм загрузки БЗ с заданной структурой в ходе диалога пользователя с системой.

Рисунков - 3, библ. наименований - 4.

УДК 681.3.06

Разработка интерфейса между языковым процессором и СУБД в генераторной системе обработки данных ГЕНСИ. Йокк В.А., Цунапуу Э.Х.-Т. - Труды Таллинского политехнического института, 1987, № 645, с. 25-32.

В современных системах обработки данных обязательным компонентом является система управления базами данных (СУБД). Использование СУБД в качестве ядра системы обработки данных требует решения проблемы разработки интерфейса между базой данных и языками обработки данных. В статье рассматривается проблема интерфейса в более общей постановке, чем в обычных системах СУБД с базовым языком.

Рисунков - 1, библ. наименований - 3.

УДК 681.3.06

О технологизации построения прикладных систем обработки данных. Выханду Л.К., Йокк В.А., Цунапуу Э.Х.-Т - Труды Таллинского политехнического института, 1987, № 645, с. 33-38.

В данной статье рассматриваются проблемы создания технологических средств системы обработки данных, которые требуют не традиционного использования системы построения трансляторов.

Рисунков - 1, библ. наименований - 4.

УДК 681.3

Статистический анализ тренда. Кривенко М.П., Мацкевич И.В., Выханду Л.К. - Труды Таллинского политехнического института, 1987, № 645, с. 39-48.

Анализ случайных процессов с изменяющимися во время наблюдения характеристиками приобретает все большее зна-

чение при автоматизированных системах научных исследований.

В данной статье предлагается один их возможных подходов, который предполагает, что изменение характеристик наблюдаемой последовательности есть неизвестная монотонная функция от номера наблюдения. Дается формализация задачи, выводятся основные теоремы и предлагается эффективный алгоритм нахождения сглаженного ряда наблюдений.

Библ. наименований - 4.

УДК 683.05

Об одном способе ускорения обновления данных в инвертированных файлах. Выханду П.Л. - Труды Таллинского политехнического института, 1987, № 645, с. 49-57.

В статье предлагается метод, использующий в инвертированных списках вместо упорядоченной цепочки бинарные деревья. Это уменьшает требуемое количество операций с $O(n/2)$ на $O(\log_2 n)$.

Рисунков - 1, библ. наименований - 3.

УДК 681.3

Применение СПТ при построении генератора отчетов. Йоонсаар Ю.Р., Ноор Ю.М. - Труды Таллинского политехнического института, 1987, № 645, с. 58-63.

В статье анализируются возможности применения СПТ при реализации языка описания отчетов. Предлагается использовать две семантические программы, связанные с одной грамматикой предшествования, что позволяет ускорить создание отчетов. В качестве иллюстрации представляется пример реализации языка ОКО на базе СПТ ЭЛМА.

Таблиц - 1, библ. наименований - 2.

УДК 51:801

Генерация сообщений о синтаксических ошибках. Рохтла Х.Х. - Труды Таллинского политехнического института, 1987, № 645, с. 64-71.

Данная статья посвящается проблемам, связанным с сообщениями о синтаксических ошибках. Во-первых, дается об-

зор работ по этой области. Во-вторых, рассматриваются принципы генерации сообщений о синтаксических ошибках и типы сообщений системы построения трансляторов ЭЛМА.

Библ. наименований - 18.

УДК 681.142

Определение трансляции АДА-ДИАНА с помощью S-атрибутных грамматик. Лепп М.В., Хейнсоо М.А. - Труды Таллинского политехнического института, 1986, № 645, с. 72-80

В данной статье представляется определение преобразования АДА-программы к виду ДИАНА-деревьев при помощи S-атрибутных грамматик. Приводятся возникающие при этом проблемы. Списываются функции - конструкторы вершин дерева, специально созданные для рассматриваемого преобразования.

Рисунков - 2, библ. наименований - 6.

УДК 681.3.06

Проблемы создания информационных систем реального времени. Микли Т.Й. - Труды Таллинского политехнического института, 1987, № 645, с. 81-89.

В статье анализируются возможности и требования к системам реального времени. Предлагаются принципы построения программного обеспечения для этих систем.

Библ. наименований - 4.

УДК 681.3

База данных для систем реального времени. Эльмик Л.Н., Роост М.Х. - Труды Таллинского политехнического института, 1987, № 645, с. 90-99

В статье рассматриваются некоторые требования для базы данных в реальном масштабе времени и особенности ввода-вывода этой системы.

Рисунков - 1, библ. наименований - 5.

УДК 681.3

О разработке технологических структур для систем реального времени. Лумберг Т.А., Распель П.У. - Труды Таллинского политехнического института, 1987, № 645, с. 100-113.

В статье рассматриваются вопросы создания технологии обработки в реальном времени, с учетом требований, предъявляемых к такой обработке. Также описывается обработка документов средствами технологического макетного обмена. Разрабатываются вопросы реализации технологических структур.

Рисунков - 3, библ. наименований - 4.

УДК 681.3.06

Система макетного обмена МАО. Бернштейн Е. - Труды Таллинского политехнического института, 1987, № 645 с. 114-117.

Ввиду широкого распространения дисплеев возникла необходимость в разработке программного обеспечения, позволяющего использовать преимущества наглядного представления информации на экране дисплея. В статье описана система макетного обмена МАО, представляющая собой набор программных средств для работы с документами на экране терминала. Система состоит из макетного редактора и макетного драйвера. В статье также приведен пример использования МАО.

УДК 681.3

Практические аспекты проектирования информационных систем. Ваппер Т., Лааст-Лаас Ю., Эльмик Л., Урвак А. - Труды Таллинского политехнического института, 1987, № 645, с. 118-125.

В статье рассматривается смешанная методика проектирования информационных систем, используемая на кафедре обработки информации Таллинского политехнического института. Методическими источниками и компонентами нашего подхода являются инфологический подход к базам данных и методика стратегического планирования BSP. В качестве инструмента сбора

и регистрации результатов системных исследований и проектирования применяется настенная техника. Представлены общие положения названной методики, дан ряд конкретных и практических рекомендаций для проведения системного анализа, для определения так называемой объект-системы и информационного моделирования ее в базе данных, а также для декомпозиции базы данных.

Библ. наименований - 5.

УДК 681.3.016

Функциональные аспекты автоматизации экономического анализа. Рензер А. - Труды Таллинского политехнического института, 1987, № 645, с. 126-133.

В статье обсуждаются проблемы автоматизации экономического анализа на уровне предприятия. Рассматривается проектирование автоматизированной инфосистемы и предлагается функциональная модель системы экономического анализа.

Библ. наименований - 2.

УДК 681.3

Один метод программирования множеств на языке АДА. Юргенсон Р.Р. - Труды Таллинского политехнического института, 1987, № 645, с. 134-140.

Статья посвящена переводу множественного типа языка Паскаль на язык Ада. Используемая методика позволяет создать на языке Ада абстрактный множественный тип данных, для значений которых применимы все известные из Паскаля операции. Совокупность объявления множественного типа и соответствующих операций оформлена в виде пакета.

Библ. наименований - 2.

УДК 681.3

Генератор гипотез для качественных признаков. Куузик Р.Э. - Труды Таллинского политехнического института, 1987, № 645, с. 141-148.

В статье предлагается новый подход для генерации гипотез из конечной совокупности качественых признаков, предла-

гаются соответствующие алгоритмы и представляются основные теоремы, доказанные в ходе разработки алгоритмов.

Библ. наименований - 7.

УДК 681.3

Пакеты прикладных программ. Лучковский Т.Ф. - Труды Таллинского политехнического института, 1987, № 645 с. 149-152.

В статье рассматриваются причины малого использования пакетов прикладных программ при создании информационных систем. Даются некоторые советы разработчикам ППП для улучшения качества пакетов.

Библ. наименований - 3.

УДК 517.934

Качественное исследование нелинейной САУ второго порядка. Геращенко-Тынисмяги С.М., Тяхт Т.К. - Труды Таллинского политехнического института, 1987, № 645, с. 153-159.

Методами качественной теории дифференциальных уравнений исследуется нелинейная САУ второго порядка с релейным уравнением. Устойчивость движения доказывается построением функции Ляпунова.

Рисунков - 3, библ. наименований - 4.

УДК 528.94

Схемы - модели взаимосвязей признаков и факторов при исследовании причин преступлений. Тынисмяги С.М., Тынисмяги Х.А. - Труды Таллинского политехнического института, 1987, № 645, с. 160-168.

Исследованы причины тяжких преступлений против личности методами математической статистики и модификацией, методом факторного анализа. В качестве статистического ряда были взяты данные из уголовных дел в Эстонской ССР.

Библ. наименований - 5.

УДК 519.4

Одно свойство знакопеременной группы A_4. Пуусемп П.А.- Труды Таллинского политехнического института, 1987, № 645, с. 169-173.

Существуют группы, которые не определяются своими полугруппами эндоморфизмов в классе всех групп. В работе доказывается, что среди указанных групп минимальный порядок имеет знакопеременная группа A_4.

Библ. наименований - 10.

Цена 1 руб. 80 коп.